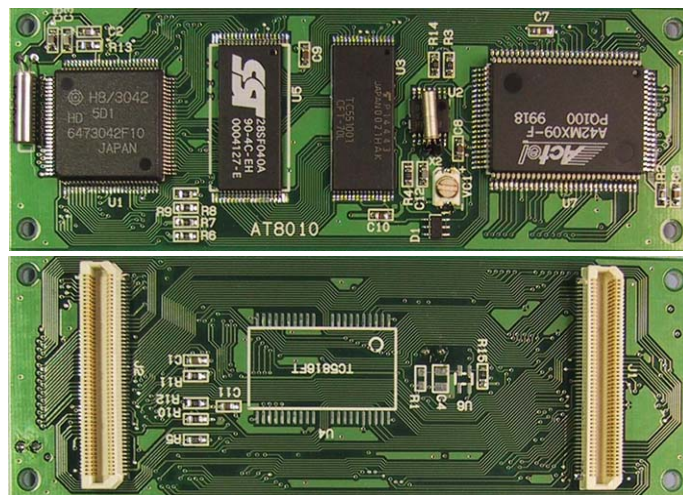
「小型マイコンモジュール」
ＡＴ８０１０
取　扱　説　明　書

【実寸大】

株式会社　エーシーティー・エルエスアイ

本ユーザーズマニュアル掲載の技術情報及び半導体のご使用につきましては以下の点にご注意願います。

1．本ユーザーズマニュアルに記載しております製品及び技術情報のうち、「外国為替及び外国貿易管理法」に基づき安全保障貿易管理関連貨物・技術に該当するものを輸出する場合、または国外に持ち出す場合は日本国政府の許可が必要です。
2．本ユーザーズマニュアルに記載された製品及び技術情報は、製品を理解していただくためのものであり、その使用に際して、当社もしくは第三者の特許権、著作権、商標権、その他の知的所有権等の権利に対する保証または実施権の許諾を意味するまた本書に記載された技術情報を使用したことにより第三者の知的所有権の権利に関わる問題が生じた場合、当社は責任を負いかねますのでご了承ください。
3．本ユーザーズマニュアルに記載しております製品及び製品仕様は、改良などのため、予告なく変更することがあります。また製造を中止する場合もありますので、ご使用に際しましては、当社または代理店に最新の情報をお問い合わせください。
4．当社は品質・信頼性の向上に努めておりますが、故障や誤動作が人命を脅かしたり、人体に危害を及ぼす恐れのある特別な品質・信頼性が要求される装置（航空宇宙機器、原子力制御システム、交通機器、輸送機器、燃焼機器、各種安全装置、生命維持関連の医療機器等）に使用される際には、必ず事前に当社にご相談ください。
5．当社は品質・信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生します。故障の結果として人身事故、火災事故、社会的な損害等を生じさせない冗長設計、延焼対策設計、誤動作防止設計等安全設計に十分ご留意ください。
誤った使用または不適切な使用に起因するいかなる損害等についても、当社は責任を負いかねますのでご了承ください。
6．本ユーザーズマニュアルに記載しております製品は、耐放射線設計はなされておりません。
7．本ユーザーズマニュアルの一部または全部を文書による当社の承諾なしで、転載または複製することを堅くお断りします。
8．本ユーザーズマニュアルに関する詳細についてのお問い合わせ、その他お気づきの点がございましたら当社または代理店までご相談ください。

2001 年 6 月

【注】
Microsoft,MS,MS-DOS,Windows,WindowsNT は米国 Microsoft 社の米国およびその他の国における登録商標です。
その他、記載されている製品名は各社の商標または登録商標です。

改訂履歴

Rev.	Date	改　訂　内　容	備　　考
1.0	00/06/26	初　版	

はじめに

この説明書は**ＡＴ８０１０**特有の機能などを記述したもので、Ｈ８マイコンに関する仕様は記述しておりません。また、オプションで用意されている、ＲＴＣ、ＮＡＮＤ型フラッシュメモリーについても同様です。ご利用になる場合はＨ８マイコンハードウェアマニュアルを入手してください。

H8 マイコンの概要

ＡＴ８０１０に搭載された、株式会社日立製作所製のマイコンＨ８／３００Ｈシリーズ（以下Ｈ８マイコン）は１６ビット×１６個の汎用レジスタをもち、１６Ｍバイトのリニアなアドレス空間を利用できます。また、周辺機能が豊富で１６ビットタイマー、プログラマブルタイミングパターンコントローラ、ウォッチドッグタイマー、シリアルコミュニケーションインタフェース、１０ビットＡ／Ｄ変換器、８ビットＤ／Ａ変換器、ＤＭＡコントローラ、リフレッシュコントローラなどを内臓しています。

機能の詳細については
株式会社日立製作所発行のマニュアルを参照してください。

・ Ｈ８／３０４２シリーズ ハードウェアマニュアル
・ Ｈ８／３００Ｈシリーズ プログラミングマニュアル ｅｔｃ．

目　　　　　次

第１章　概　要

ＡＴ８０１０は１６ビットマイコン「Ｈ８／３０４２」を搭載したマイコンモジュールです。標準仕様で１ＭビットＳＲＡＭ及び、４ＭビットフラッシュＲＯＭ（内ユーザー領域は３８４Ｋバイト）を搭載し、オプションで株式会社リコー製リアルタイム・クロック「RS5C317A」及び、ＮＡＮＤ型フラッシュメモリー（２Ｍ、４Ｍ、８Ｍバイト）が搭載可能となっています。

Ｈ８マイコンの豊富な機能をそのまま利用できるように１００ピンの峡ピッチコネクタを２個使用し、すべて機能ポートを外部に直接接続できるようにしてあります。それに加え、フィールド・プログラマブル・ゲート・アレイ（ＦＰＧＡ）を搭載することによりＩ／Ｏポートの拡張を行い、Ｈ８の機能ポートを十分に利用できるようにしてあります。
また、アドレス及びデータバスは弊社のマイコンモジュールシリーズで共通化されたピン配列で設計してありますので、機能端子の配列に注意して利用すれば、CPU のパワーアップなどが可能です。１９９９年１月現在では日立製作所製「SH7708」を搭載したマイコンモジュール「AT8011」と共通仕様になっています。（但し、電源電圧が違います）

ユーザーが開発したプログラムはRS-232C通信ターミナルソフト（XMODEMプロトコル）で転送、書き換えが可能になっています。

■使用上の注意点
Ｈ８のＮＭＩはＦＰＧＡ内にてマスク可能になっております。従って、ＮＭＩ　ＲＴＣのアラーム、定周期割込および外部割込の選択が可能です。（４．１項 割込制御レジスタ参照）

１．１ 構成

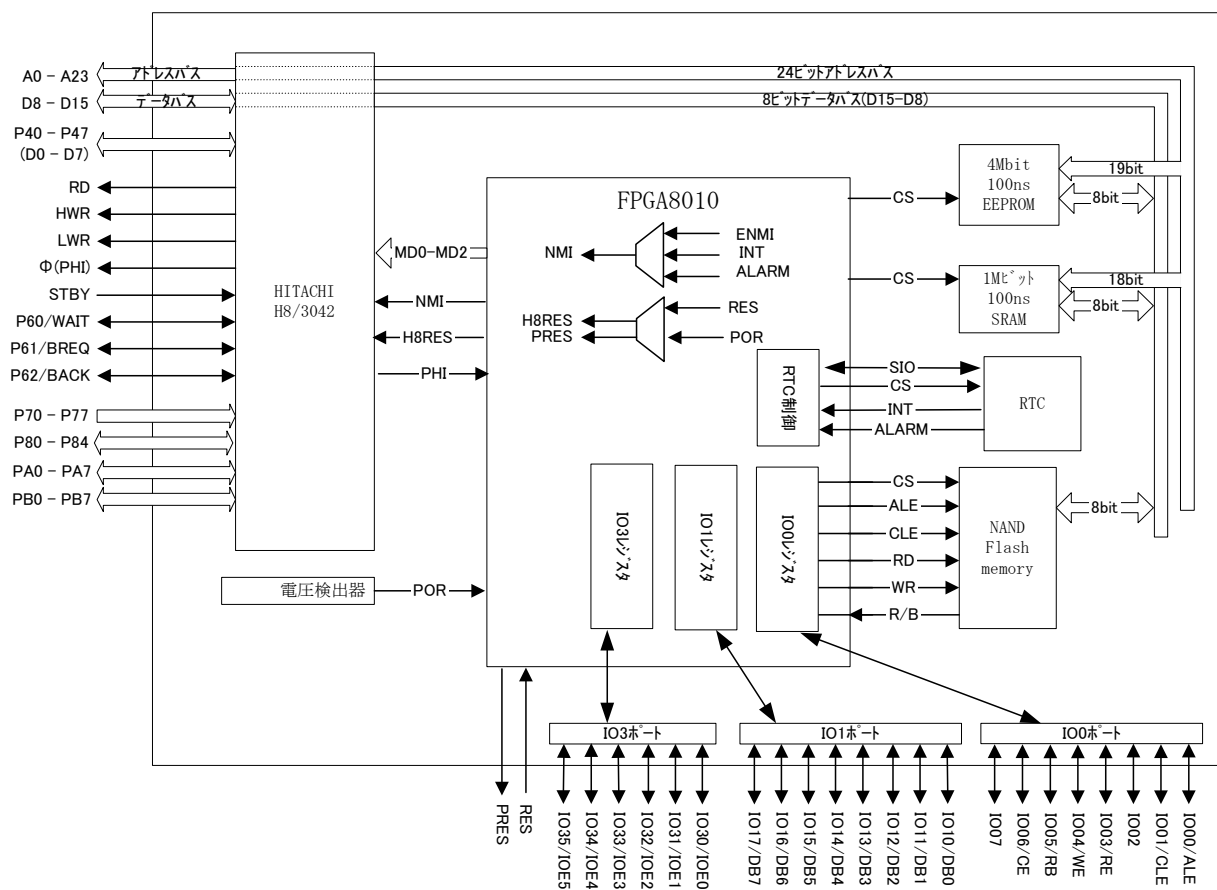
ＡＴ８０１０には標準で１ＭビットのＳＲＡＭと４ＭビットのフラッシュＲＯＭが搭載されています。

フラッシュＲＯＭの一部はシステムで使用していますので、ユーザーが利用できるプログラム領域は３８４Ｋバイトになっています。システムのバージョンアップなどをしない場合は、５１２Ｋバイトフルに利用可能となります（システムのバージョンアップを参照）。

オプションで搭載されるリアルタイム・クロックはアラーム機能付きで、その出力信号はＦＰＧＡを経由しＨ８のＮＭＩ端子に接続されていますので低消費電力状態からの復帰にも利用できます。詳細については後章の機能レジスタで説明します。

ＮＡＮＤ型フラッシュメモリーは株式会社東芝製 「ＴＣ５８３２ＦＴ」または「ＴＣ５８６４ＦＴ」相当品を搭載します。このメモリーはそれぞれ２Ｍ、４Ｍ、８Ｍバイトのメモリー容量を持ち、主にストレージメモリーとして利用します（プログラムメモリーとしては利用できません）。

ＡＴ８０１０の構成図を図１に示します。

図 1　AT8010 構成図

１.２ 概 観

基板外形及び嵌合コネクタのプリント基板パターン図を以下に示します。

図 2　基板寸法図

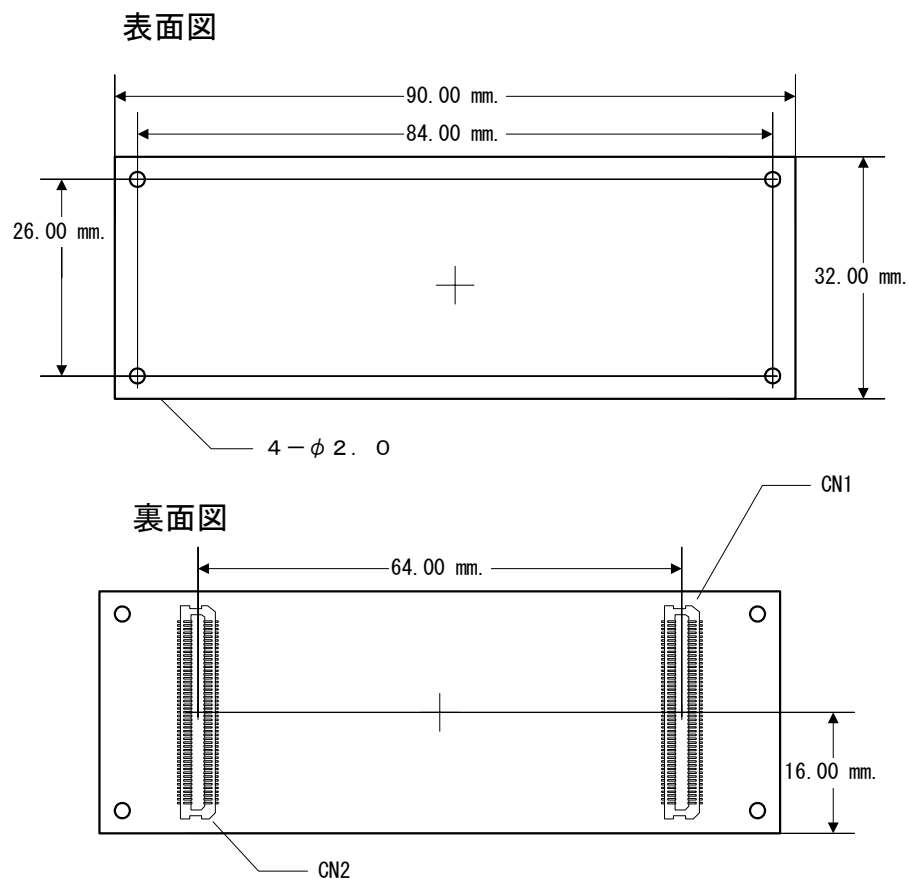
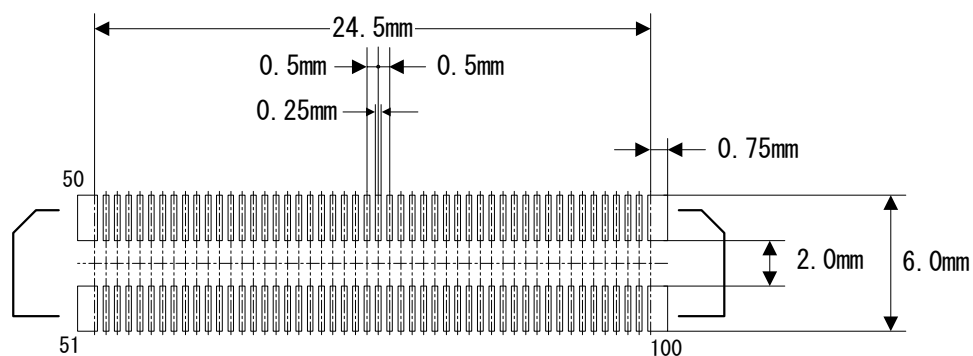



図 3 嵌合コネクタパターン図

１.３ 端子構成

ＡＴ８０１０はＨ８の端子を可能な限り外部へ出すことを考慮し、外部接続コネクタに松下電工株式会社製 峡ピッチ多極コネクタ 「ＡＸＫ５Ｓ０００４５」を採用しました。これに嵌合するコネクタは同社製 「ＡＸＫ６Ｓ００５４５Ｐ」となります。概観図にプリント基板パターン図を添付しました。端子番号もその図を参照してください。
以下に AT8010 の端子表を示します。信号名に＊がついているものは H8 マイコンに直接接続されています。

CN1 端子表

端 子 番 号	端 子 名	説 明
1, 50, 51, 100	VDD	電源（＋5V）
10, 19,26, 32, 41, 60,69, 75, 82, , 91	GND	グランド
2 – 9, 11 – 18, 33 – 40, 70 – 72, 76, 77, 80	NC	何も接続しないでください。
20	IO30/$\overline{\text{IOE0}}$	I/Oまたはｱﾄﾞﾚｽﾃﾞｺｰﾄﾞ信号(FFF000 H – FFF0FF H)
21	IO31/$\overline{\text{IOE1}}$	I/Oまたはｱﾄﾞﾚｽﾃﾞｺｰﾄﾞ信号(FFF100 H – FFF1FF H)
22	IO32/$\overline{\text{IOE2}}$	I/Oまたはｱﾄﾞﾚｽﾃﾞｺｰﾄﾞ信号(FFF200 H – FFF2FF H)
23	IO33/$\overline{\text{IOE3}}$	I/Oまたはｱﾄﾞﾚｽﾃﾞｺｰﾄﾞ信号(FFF300 H – FFF3FF H)
24	IO34/$\overline{\text{IOE4}}$	I/Oまたはｱﾄﾞﾚｽﾃﾞｺｰﾄﾞ信号(FFF400 H – FFF4FF H)
25	IO35/$\overline{\text{IOE5}}$	I/Oまたはｱﾄﾞﾚｽﾃﾞｺｰﾄﾞ信号(FFF500 H – FFF5FF H)
42	*A23/TIOCA1/TP4/PA4	機能ﾎﾟｰﾄ(ﾕｰｻﾞｰﾓｰﾄﾞ3～6ではｱﾄﾞﾚｽﾊﾞｽ)
43	*A22/TIOCB1/TP5/PA5	機能ﾎﾟｰﾄ(ﾕｰｻﾞｰﾓｰﾄﾞ3～6ではｱﾄﾞﾚｽﾊﾞｽ)
44	*A21/TIOCA2/TP6/PA6	機能ﾎﾟｰﾄ(ﾕｰｻﾞｰﾓｰﾄﾞ3～6ではｱﾄﾞﾚｽﾊﾞｽ)
45	*A20/TIOCB2/TP7/PA7	機能ﾎﾟｰﾄ(ﾕｰｻﾞｰﾓｰﾄﾞ3～6ではｱﾄﾞﾚｽﾊﾞｽ)
46	*A19	ｱﾄﾞﾚｽﾊﾞｽ
47	*A18	ｱﾄﾞﾚｽﾊﾞｽ
48	*A17	ｱﾄﾞﾚｽﾊﾞｽ
49	*A16	ｱﾄﾞﾚｽﾊﾞｽ
52	*A15	ｱﾄﾞﾚｽﾊﾞｽ
53	*A14	ｱﾄﾞﾚｽﾊﾞｽ
54	*A13	ｱﾄﾞﾚｽﾊﾞｽ
55	*A12	ｱﾄﾞﾚｽﾊﾞｽ
56	*A11	ｱﾄﾞﾚｽﾊﾞｽ
57	*A10	ｱﾄﾞﾚｽﾊﾞｽ

端 子 番 号	端　子　名	説　　　明
58	*A9	ｱﾄﾞﾚｽﾊﾞｽ
59	*A8	ｱﾄﾞﾚｽﾊﾞｽ
61	*A7	ｱﾄﾞﾚｽﾊﾞｽ
62	*A6	ｱﾄﾞﾚｽﾊﾞｽ
63	*A5	ｱﾄﾞﾚｽﾊﾞｽ
64	*A4	ｱﾄﾞﾚｽﾊﾞｽ
65	*A3	ｱﾄﾞﾚｽﾊﾞｽ
66	*A2	ｱﾄﾞﾚｽﾊﾞｽ
67	*A1	ｱﾄﾞﾚｽﾊﾞｽ
68	*A0	ｱﾄﾞﾚｽﾊﾞｽ
73	*P63/$\overline{AS}$	ポート / ｱﾄﾞﾚｽｽﾄﾛｰﾌﾞ信号出力
74	*P60/$\overline{WAIT}$	ポート / ｳｴｲﾄ信号入力
78	*$\overline{LWR}$	D0～D8書込ｽﾄﾛｰﾌﾞ信号
79	*$\overline{HWR}$	D8～D15書込ｽﾄﾛｰﾌﾞ信号
81	*$\overline{RD}$	ﾊﾞｽ読出信号出力
83	*D7/P47	ﾎﾟｰﾄ / ﾃﾞｰﾀﾊﾞｽ
84	*D6/P46	ﾎﾟｰﾄ / ﾃﾞｰﾀﾊﾞｽ
85	*D5/P45	ﾎﾟｰﾄ / ﾃﾞｰﾀﾊﾞｽ
86	*D4/P44	ﾎﾟｰﾄ / ﾃﾞｰﾀﾊﾞｽ
87	*D3/P43	ﾎﾟｰﾄ / ﾃﾞｰﾀﾊﾞｽ
88	*D2/P42	ﾎﾟｰﾄ / ﾃﾞｰﾀﾊﾞｽ
89	*D1/P41	ﾎﾟｰﾄ / ﾃﾞｰﾀﾊﾞｽ
90	*D0/P40	ﾎﾟｰﾄ / ﾃﾞｰﾀﾊﾞｽ
92	*D15	ﾃﾞｰﾀﾊﾞｽ（AT8010内部ではD7）
93	*D14	ﾃﾞｰﾀﾊﾞｽ（AT8010内部ではD6）
94	*D13	ﾃﾞｰﾀﾊﾞｽ（AT8010内部ではD5）
95	*D12	ﾃﾞｰﾀﾊﾞｽ（AT8010内部ではD4）
96	*D11	ﾃﾞｰﾀﾊﾞｽ（AT8010内部ではD3）
97	*D10	ﾃﾞｰﾀﾊﾞｽ（AT8010内部ではD2）
98	*D9	ﾃﾞｰﾀﾊﾞｽ（AT8010内部ではD1）
99	*D8	ﾃﾞｰﾀﾊﾞｽ（AT8010内部ではD0）

CN2 端子表

端 子 番 号	端　子　名	説　　　明
1, 50, 51,100	VDD	電源（＋5V）
10, 19, 26, 32, 41, 60, 69, 75, 82, , 91	GND	グランド
49, 52	VBU	RTC用ﾊﾞｯｸｱｯﾌﾟ電源接続端子（1V～5 V）
15,16,23-25,27,35-38, 42-48,55-57, 59	NC	何も接続しないでください
2	*TIOCA3/TP8/PB0	機能ﾎﾟｰﾄ
3	*TIOCB3/TP9/PB1	機能ﾎﾟｰﾄ
4	*TIOCA4/TP10/PB2	機能ﾎﾟｰﾄ
5	*TIOCB4/TP11/PB3	機能ﾎﾟｰﾄ
6	*TOCXA4/TP12/PB4	機能ﾎﾟｰﾄ
7	*TOCXAB4/TP13/PB5	機能ﾎﾟｰﾄ
8	*$\overline{DREQ0}$/TP14/PB6	機能ﾎﾟｰﾄ
9	*$\overline{ADTRG}$/$\overline{DREQ1}$/TP15/PB7	機能ﾎﾟｰﾄ
11	*PA0/TP0/$\overline{TEND0}$/TCLKA	機能ﾎﾟｰﾄ
12	*PA1/TP1/$\overline{TEND1}$/TCLKB	機能ﾎﾟｰﾄ
13	*PA2/TP2/TIOCA0/TCLKC	機能ﾎﾟｰﾄ
14	*PA3/TP3/TIOCB0/TCLKD	機能ﾎﾟｰﾄ
17	IO20	入出力ﾎﾟｰﾄ
18	IO17/DB7	入出力ﾎﾟｰﾄまたは拡張ﾃﾞｰﾀﾊﾞｽ
20	IO16/DB6	入出力ﾎﾟｰﾄまたは拡張ﾃﾞｰﾀﾊﾞｽ
21	IO15/DB5	入出力ﾎﾟｰﾄまたは拡張ﾃﾞｰﾀﾊﾞｽ
22	IO14/DB4	入出力ﾎﾟｰﾄまたは拡張ﾃﾞｰﾀﾊﾞｽ
23	IO13/DB3	入出力ﾎﾟｰﾄまたは拡張ﾃﾞｰﾀﾊﾞｽ
24	IO12/DB2	入出力ﾎﾟｰﾄまたは拡張ﾃﾞｰﾀﾊﾞｽ
25	IO11/DB1	入出力ﾎﾟｰﾄまたは拡張ﾃﾞｰﾀﾊﾞｽ
27	IO10/DB0	入出力ﾎﾟｰﾄまたは拡張ﾃﾞｰﾀﾊﾞｽ
33	*P61/$\overline{BREQ}$	機能ﾎﾟｰﾄ
34	*P62/$\overline{BACK}$	機能ﾎﾟｰﾄ
39	*$\overline{STBY}$	ｽﾀﾝﾊﾞｲ端子
40	NMI	FPGAを介しH8マイコンのNMI端子に接続
53	$\overline{RES}$	ﾘｾｯﾄ入力
54	$\overline{PRES}$	外部ﾘｾｯﾄ信号出力
58	*PHI	ｸﾛｯｸ出力（9.8304MHz）
59	32KOUT	RTC調整用（32KHz出力）

端 子 番 号	端　子　名	説　　　明
61	IO07	入出力ﾎﾟｰﾄ
62	IO06/FMCS	入出力ﾎﾟｰﾄまたはNAND型ﾌﾗｯｼｭﾒﾓﾘｰのﾁｯﾌﾟ選択信号
63	IO05/FMRB	入出力ﾎﾟｰﾄまたはNAND型ﾌﾗｯｼｭﾒﾓﾘｰのﾋﾞｼﾞｰ信号
64	IO04/FMWE	入出力ﾎﾟｰﾄまたはNAND型ﾌﾗｯｼｭﾒﾓﾘｰの書込信号
65	IO03/FMRE	入出力ﾎﾟｰﾄまたはNAND型ﾌﾗｯｼｭﾒﾓﾘｰの読出信号
66	IO02	入出力ﾎﾟｰﾄ
67	IO01/FMCLE	入出力ﾎﾟｰﾄまたはNAND型ﾌﾗｯｼｭﾒﾓﾘｰのｱﾄﾞﾚｽ選択信号
68	IO00/FMALE	入出力ﾎﾟｰﾄまたはNAND型ﾌﾗｯｼｭﾒﾓﾘｰのｺﾏﾝﾄﾞ選択信号
70	*Avdd	ｱﾅﾛｸﾞ電源
71	*Avref	ｱﾅﾛｸﾞﾘﾌｧﾚﾝｽ
72	*P70/AN0	機能ﾎﾟｰﾄ
73	*P71/AN1	機能ﾎﾟｰﾄ
74	*P72/AN2	機能ﾎﾟｰﾄ
76	*P73/AN3	機能ﾎﾟｰﾄ
77	*P74/AN4	機能ﾎﾟｰﾄ
78	*P75/AN5	機能ﾎﾟｰﾄ
79	*P76/AN6/DA0	機能ﾎﾟｰﾄ
80	*P77/AN7/DA1	機能ﾎﾟｰﾄ
81	AVSS	ｱﾅﾛｸﾞｸﾞﾗﾝﾄﾞ
83	*P80/$\overline{RFSH}$/$\overline{IRQ0}$	機能ﾎﾟｰﾄ
84	*P81/$\overline{CS3}$/$\overline{IRQ1}$	機能ﾎﾟｰﾄ
85	*P82/$\overline{CS2}$/$\overline{IRQ2}$	機能ﾎﾟｰﾄ
86	*P83/$\overline{CS1}$/$\overline{IRQ3}$	機能ﾎﾟｰﾄ
87	*P84/$\overline{CS0}$	機能ﾎﾟｰﾄ
88	*$\overline{IRQ5}$/SCK1/P95	機能ﾎﾟｰﾄ
89	*$\overline{IRQ4}$/SCK0/P94	機能ﾎﾟｰﾄ
90	*RXD1/P93	機能ﾎﾟｰﾄ
92	*RXD0/P92	ﾌﾟﾛｸﾞﾗﾑ書き換え時の受信ﾎﾟｰﾄ
93	*TXD0/P90	ﾌﾟﾛｸﾞﾗﾑ書き換え時の送信ﾎﾟｰﾄ
94	*TXD1/P91	機能ﾎﾟｰﾄ
95	CLKC	RTC調整用ｸﾛｯｸ出力ｲﾈｰﾌﾞﾙ端子
96	M3	ﾓｰﾄﾞ設定端子
97	M2	ﾓｰﾄﾞ設定端子
98	M1	ﾓｰﾄﾞ設定端子
99	M0	ﾓｰﾄﾞ設定端子

１．４ アドレスマップ

ＡＴ８０１０のモード別アドレスマップを以下に示します。

図 4　アドレスマップ１

図 5　アドレスマップ２

AT8010 Memory map 2
AT8010 USER MODE 2 (H8/3042 MODE 3)
USER ROM 384KB
EXTERN ADDRESS
STATIC RAM 128KB
EXTERN ADDRESS
PERIPHERAL DEVICE CONTROL
AT8010 機能ﾚｼﾞｽﾀ
RTC ﾚｼﾞｽﾀ
EXTERN ADDRESS
内臓RAM
内臓I/Oﾚｼﾞｽﾀ
AT8010 USER MODE 3 (H8/3042 MODE 3)
USER ROM 384KB
SYSTEM ROM 128KB
EXTERN ADDRESS
STATIC RAM 128KB
EXTERN ADDRESS
PERIPHERAL DEVICE CONTROL
AT8010 機能ﾚｼﾞｽﾀ
RTC ﾚｼﾞｽﾀ
EXTERN ADDRESS
内臓RAM
内臓I/Oﾚｼﾞｽﾀ
AT8010 USER MODE 4 (H8/3042 MODE 3)
USER ROM 384KB
EXTERN ADDRESS
STATIC RAM 128KB
EXTERN ADDRESS
PERIPHERAL DEVICE CONTROL
AT8010 機能ﾚｼﾞｽﾀ
RTC ﾚｼﾞｽﾀ
SYSTEM RESERVE
内臓RAM
内臓I/Oﾚｼﾞｽﾀ
AT8010 USER MODE 5 (H8/3042 MODE 3)
USER ROM 384KB
SYSTEM ROM 128KB
EXTERN ADDRESS
STATIC RAM 128KB
EXTERN ADDRESS
PERIPHERAL DEVICE CONTROL
AT8010 機能ﾚｼﾞｽﾀ
RTC ﾚｼﾞｽﾀ
SYSTEM RESERVE
内臓RAM
内臓I/Oﾚｼﾞｽﾀ

第２章　動作モード

ＡＴ８０１０はユーザーのプログラムが動作するユーザーモードとユーザーのプログラムの書き換えなどを行うシステムモードに大別されます。モードの切り替えは端子ＭＤ０～ＭＤ３の４本のモード端子で行い、６つのユーザーモードと５つのシステムモードに切り替わります。

H8マイコンにも動作モードがありますが、ＡＴ８０１０ではH8のモード１、３、５の３つのモードいずれかのモードで動作します。ユーザーモードでは、H8のモード１または３で、システムモードはＨ８のモード５または１で動作します。

モード端子とモードの設定を表１に示します。

表 1 モード設定表

	モード端子状態(1 で High ﾚﾍﾞﾙ)				H8 ﾓｰﾄﾞ	
	MD3	MD2	MD1	MD0		
ユーザーモード	1	1	1	1	1	ﾕｰｻﾞｰﾓｰﾄﾞ 0
	1	1	1	0	1	ﾕｰｻﾞｰﾓｰﾄﾞ 1
	1	1	0	1	3	ﾕｰｻﾞｰﾓｰﾄﾞ 2
	1	1	0	0	3	ﾕｰｻﾞｰﾓｰﾄﾞ 3
	1	0	1	1	3	ﾕｰｻﾞｰﾓｰﾄﾞ 4
	1	0	1	0	3	ﾕｰｻﾞｰﾓｰﾄﾞ 5
	1	0	0	1	-	未使用
	1	0	0	0	-	未使用
システムモード	0	1	1	1	5	ｼｽﾃﾑﾓｰﾄﾞ 0
	0	1	1	0	5	ｼｽﾃﾑﾓｰﾄﾞ 1
	0	1	0	1	1	ｼｽﾃﾑﾓｰﾄﾞ 2
	0	1	0	0	1	ｼｽﾃﾑﾓｰﾄﾞ 3
	0	0	1	1	-	未使用
	0	0	1	0	-	未使用
	0	0	0	1	-	未使用
	0	0	0	0	5	ｼｽﾃﾑﾌﾟﾛｸﾞﾗﾑ書換ﾓｰﾄﾞ

モード端子は開放状態で１（High ﾚﾍﾞﾙ）になっています。

第３章　ＡＴ８０１０機能ポート

ＡＴ８０１０には２３本の I/O 端子があり、入出力及び機能ポートとして利用できます。各機能ポートは初期状態で入力ポートとして設定されており、機能選択レジスタで機能を指定します。レジスタは各機能毎に４分割されており、それぞれ、ＮＡＮＤ型フラッシュメモリー制御（８ビット)、８ビットバス（８ビット)、デバイス選択信号出力（６ビット）となっています。また、各機能毎に、機能設定レジスタ、方向設定レジスタ、データレジスタが用意されています。

３．１ 入出力ポート

ＡＴ８０１０の各機能ポートは初期状態で入力ポートとして設定されています。
機能設定レジスタの各ビットが０の時、入出力ポートとして機能します。
入出力方向設定レジスタのそれぞれのビットが０の時、入力ポート、１の時、出力ポートとなります。
設定された各ポートの状態は、データレジスタに反映されます。入力に設定されたポートがＨｉｇｈレベルの場合、データレジスタのビットが１、Ｌｏｗレベルの場合、０が読み出されます。出力に設定されたポートはデータレジスタのビットに１を書き込むとＨｉｇｈレベルが出力され、０を書き込むとＬｏｗレベルが出力されます。
入力に設定されたポートに対応するデータレジスタビットに書き込みを行っても無効となります。

３．２ 外部ﾃﾞﾊﾞｲｽ選択信号

　ＡＴ８０１０に搭載されたＦＰＧＡはＨ８マイコンの２４ビットアドレス出力信号（Ａ０～Ａ２３）により、アドレスデコード信号を作成し出力します。この信号はＩＯ３０～ＩＯ３５端子に割り当てられており、機能レジスタでＩＯ端子として利用するか、デコード信号として利用するかを選択できます。
　H8/3042 には１Ｍ及び１６Ｍバイトのアドレスモードがあり、それぞれのモードでデバイスのマッピングが異なります。以下の表に各デバイスを選択するアドレスを記します。

ＡＴ８０１０モード０～３のマッピング（H8 モード１）

アドレス	端　子　名
0FF000 h～0FF0FF h	IO30 / $\overline{\text{IOE0}}$
0FF100 h～0FF1FF h	IO31 / $\overline{\text{IOE1}}$
0FF200 h～0FF2FF h	IO32 / $\overline{\text{IOE2}}$
0FF300 h～0FF3FF h	IO33 / $\overline{\text{IOE3}}$
0FF400 h～0FF4FF h	IO34 / $\overline{\text{IOE4}}$
0FF500 h～0FF5FF h	IO35 / $\overline{\text{IOE5}}$

ＡＴ８０１０モード６，７のマッピング（H8 モード３）

アドレス	端　子　名
FFF000 h～FFF0FF h	IO30 / $\overline{\text{IOE0}}$
FFF100 h～FFF1FF h	IO31 / $\overline{\text{IOE1}}$
FFF200 h～FFF2FF h	IO32 / $\overline{\text{IOE2}}$
FFF300 h～FFF3FF h	IO33 / $\overline{\text{IOE3}}$
FFF400 h～FFF4FF h	IO34 / $\overline{\text{IOE4}}$
FFF500 h～FFF5FF h	IO35 / $\overline{\text{IOE5}}$

　これらの機能を使用するには機能レジスタの「ＩＯ３ＳＬ」を設定します。
ＩＯ３ＳＬレジスタのビット０からビット５はそれぞれ$\overline{\text{ＩＯＥ０}}$から$\overline{\text{ＩＯＥ５}}$の設定になっており、ビットを１にすることにより、アドレスデコード信号の出力になります。

注　意

ＡＴ８０１０のモード ３～６を利用する場合、以下の事に注意してください。
H8/3042 のモード３は初期状態でＡ２０からＡ２３端子は出力状態になっていません。
機能ポートを出力に設定してください。
また、

３．３ ８ビットバス

CPUのバスとは独立した８ビットの入出力データバスとして利用可能なＩＯ端子を用意してあります（ＩＯ１０～ＩＯ１７端子）。 前述の$\overline{\text{IOE}}$信号を選択し$\overline{\text{IOE}}$の出力範囲でデータバスとして利用します。

８ビットバス機能レジスタ（ＩＯ１ＳＬ）のビット７（ＢＵＳ）を１に設定し、ビット０からビット２の３ビットで対応させるアドレス範囲を選択します。

３．４ ＮＡＮＤ型フラッシュメモリー制御信号

オプションで搭載可能なＮＡＮＤ型フラッシュメモリーは通常メモリーと異なり、アドレスバスが在りません。また、読み出し、書き込み、消去などはコマンドを用い実行します。
フラッシュメモリーは上記機能を実現するために特殊な端子があり、「ＡＴ８０１０」はこの端子を制御するためのレジスタを用意してあります。
この端子はＩＯ００～ＩＯ０７に割り当てられており、入出力ポートまたは機能端子として選択可能になっています。

３．５ 割り込み制御

ＡＴ８０１０はオプションで搭載される RTC のアラーム、インターラプト信号及び外部信号を取り込み H8 の NMI 端子に出力する機能があります。
Ｈ８マイコンの端子名は“ＮＭＩ”ですが、ＡＴ８０１０では、それぞれの信号のマスク、入力エッジの切り替えが可能になっています。（入力エッジ切り替えは外部入力のみとする）

３．６ リセット制御

外部リセット信号、Ｈ８リセット出力、電圧検出器の出力を管理しシステムにリセット信号を出力します。
外部からの信号（$\overline{\text{RES}}$端子）は AT8010 の回路上でプルアップされており、Ｌｏにすることによりｈ８マイコン及び周辺回路をリセットします（PRES 端子へ出力）。
H8 から出力される信号$\overline{\text{RESO}}$端子はｳｫｯﾁﾄﾞｯｸﾀｲﾏｰの出力で、通常は High 状態となっており、内臓周辺機器及び$\overline{\text{ＰＲＥＳ}}$端子へ出力されます。

電圧検出器の信号は電源電圧が約４．８V を切ると出力され、Ｈ８マイコン、ＦＰＧＡ、内臓周辺機器、$\overline{\text{ＰＲＥＳ}}$端子へ出力されます。
$\overline{\text{ＰＲＥＳ}}$端子及びＨ８マイコンへのリセット出力は H8 マイコンのクロックで３２クロック以上の Low 区間を確保し出力されます。
また、$\overline{\text{ＰＲＥＳ}}$端子は機能レジスタで任意にＬｏｗ出力できるようになっています。

第４章　機能レジスタ

機能レジスタとは、前項で説明した機能ポートの設定、動作モードの判定など行う、ユーザーインターフェースを示します。ユーザーはこのレジスタでＡＴ８０１０の機能を設定します。
以下に機能レジスタの一覧を示します。

機能レジスタ一覧

ｱﾄﾞﾚｽ	ﾚｼﾞｽﾀ名	機　　　　能
FF600 h	NMIC	割り込み制御
FF601 h	IO0D	IO0 ﾎﾟｰﾄﾃﾞｰﾀﾚｼﾞｽﾀ
FF602 h	IO0DI	IO0 ﾎﾟｰﾄ方向設定ﾚｼﾞｽﾀ
FF603 h	IO0SL	IO0 ﾎﾟｰﾄ機能設定ﾚｼﾞｽﾀ（ﾌﾗｯｼｭﾒﾓﾘｰ制御ﾚｼﾞｽﾀ）
FF604 h	FMDT	ﾌﾗｯｼｭﾒﾓﾘｰﾃﾞｰﾀﾊﾞｽ
FF605 h	IO1D	IO1 ﾎﾟｰﾄﾃﾞｰﾀﾚｼﾞｽﾀ
FF606 h	IO1DI	IO1 ﾎﾟｰﾄ方向設定ﾚｼﾞｽﾀ
FF607 h	IO1SL	IO1 ﾎﾟｰﾄ機能設定ﾚｼﾞｽﾀ（ﾊﾞｽ制御ﾚｼﾞｽﾀ１）
FF60B h	IO3D	IO3 ﾎﾟｰﾄﾃﾞｰﾀﾚｼﾞｽﾀ
FF60C h	IO3DI	IO3 ﾎﾟｰﾄ方向設定ﾚｼﾞｽﾀ
FF60D h	IO3SL	IO3 ﾎﾟｰﾄ機能設定ﾚｼﾞｽﾀ（IOE 制御ﾚｼﾞｽﾀ）
FF60E h	WDTC	ｳｵｯﾁﾄﾞｯｸﾀｲﾏｰ設定ﾚｼﾞｽﾀ（ｵﾌﾟｼｮﾝの RTC 搭載時使用可能）
FF60F h	SYSCTL	ｼｽﾃﾑ制御ﾚｼﾞｽﾀ

機能レジスタビット名称

ﾚｼﾞｽﾀ名	b7	b6	b5	b4	b3	b2	b1	b0
NMIC	NMIS	RTAE	RTIE	NMIE		RTAG	RTIG	NMIG
IO0D	IO07	IO06	IO05	IO04	IO03	IO02	IO01	IO00
IO0DI	IOD07	IOD06	IOD05	IOD04	IOD03	IOD02	IOD01	IOD00
IO0SL	FME							
FMDB	D7	D6	D5	D4	D3	D2	D1	D0
IO1D	IO17	IO16	IO15	IO14	IO13	IO12	IO11	IO10
IO1DI	IOD17	IOD16	IOD15	IOD14	IOD13	IOD12	IOD11	IOD10
IO1SL	BUS					SL2	SL1	SL0
IO3D			IO35	IO34	IO33	IO32	IO31	IO30
IO3DI			IOD35	IOD34	IOD33	IOD32	IOD31	IOD30
IO3SL		IOE6E	IOE5E	IOE4E	IOE3E	IOE2E	IOE1E	IOE0E
WDTC	RST							WDTE
SYSCTL	PRES	MDC	RES	HAE	MD3	MD2	MD1	MD0

４．１ 割込制御レジスタ（ＮＭＩＣ）

　ＡＴ８０１０にオプションで搭載するリアルタイムクロックのアラーム出力および定周期出力と、コネクタのＮＭＩ端子は、ＦＰＧＡを介し、Ｈ８マイコンのＮＭＩ端子に接続されています。
このレジスタはＨ８マイコンのＮＭＩ端子に出力する信号を制御します。
Ｈ８のＮＭＩに接続された端子は通常Ｈｉｇｈレベルですので、Ｈ８マイコンのＮＭＩエッジ選択は初期値の立ち下がりエッジに設定してください。
また、ＡＴ８０１０に搭載されるＲＴＣはモードによって、アラームまたは定周期パルスの出力が自動復帰しない場合があります。
この場合、ＲＴＡＧ、ＲＴＩＧに０を描きこむと、再びＮＭＩが発生します。
ＲＴＣのレジスタでアラームまたは定周期パルス出力をクリアしてから実行してください。

	b7	b6	b5	b4	b3	b2	b1	B0
NMIC	NMIS	RTAE	RTIE	NMIE		RTAG	RTIG	NMIG
初期値	0	0	0	0	0	0	0	0
方　向	R/W	R/W	R/W	R/W		R/W	R/W	R/W

■ＮＭＩＳビット（ビット７）
　外部端子（ＮＭＩ端子）での、割込を発生させるエッジを選択します。０で立ち下がり、１で立ち上がりエッジになります。

■ＲＴＡＥビット（ビット６）
　ＲＴＣのアラーム割込をマスクします。０で禁止、１で許可。

■ＲＴＩＥビット（ビット５）
　ＲＴＣの定周期割込をマスクします。０で禁止、１で許可。

■ＮＭＩＥビット（ビット４）
　外部端子（ＮＭＩ端子）の割込をマスクします。０で禁止、１で許可。

■ＲＴＡＧビット（ビット２）
　ＲＴＣのアラーム割込が発生したことを通知します。ＲＴＣのアラーム割込が発生すると、本ビットが１となります。本ビットは自動的にクリアされませんので、０を書き込んでクリアしてください。クリアしなくても割込みは発生します。

■ＲＴＩＧビット（ビット１）
　ＲＴＣの定周期割込が発生したことを通知します。ＲＴＣの定周期割込が発生すると、本ビットが１となります。本ビットは自動的にクリアされませんので、０を書き込んでクリアしてください。クリアしなくても割込みは発生します。

■ＮＭＩＧビット（ビット０）
　外部端子（ＮＭＩ端子）が変化したことを通知します。外部端子が変化（ＮＭＩＳビットで設定されたエッジの方向により検出する変化）すると、本ビットが１になります。本ビットは自動的にクリアされませんので、０を書き込んでクリアしてください。クリアしなくても割込みは発生します。

　ＲＴＡＧ、ＲＴＩＧ、ＮＭＩＧビットはそれぞれのマスクビットでマスクしても、それぞれの入力に変化があればセットされます。
利用する場合マスク解除後、一度クリアしたほうが良いでしょう。

４．２ ＩＯ０ポート

■ ＩＯ０ポートデータレジスタ（ＩＯ０Ｄ）

ＩＯ０ポートデータレジスタはＩＯ００からＩＯ０７端子の状態をモニターしたり、端子の状態を設定するレジスタです。後述のＩＯ０ＤＩ方向設定レジスタまたはＩＯ０ＳＬ機能設定レジスタにより入出力の方向が決定します。また、このポートはオプションのＮＡＮＤ型フラッシュメモリーの制御端子にもなりますので、フラッシュメモリーの制御にも使用します。

	b7	b6	b5	b4	b3	b2	b1	b0
IO0D	IO07	IO06	IO05	IO04	IO03	IO02	IO01	IO00
初期値	-	-	-	-	-	-	-	-
方　向	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ポートの設定が出力に設定されている場合、本レジスタのポートに対応するビットをセットする（１にする）とＨｉｇｈレベルを出力し、クリアする（0にする）とＬｏｗレベルを出力します。ポートの設定が入力に設定されている場合、本レジスタのポートに対応するビットを読み出すと、ポートに与えられた電位がＨｉｇｈの場合１が読み出され、Ｌｏｗの場合０が読み出されます。

■ ＩＯ０ポート方向設定レジスタ（ＩＯ０ＤＩ）

ＩＯ０ポート方向設定レジスタはＩＯ０ポートの入出力方向を指定するレジスタです。１端子に付き１ビットが対応しています。このレジスタの設定はＩＯ０ポート機能設定レジスタのＦＭＥビットが１の場合無効になり、フラッシュメモリーのデータバスに割り当てられます。

	b7	b6	b5	b4	b3	b2	b1	b0
IO0DI	IOD07	IOD06	IOD05	IOD04	IOD03	IOD02	IOD01	IOD00
初期値	0	0	0	0	0	0	0	0
方　向	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ＦＭＥビットが０の場合、本レジスタのポートに対応するビットを１に設定するとそのポートは出力ポートになり、０を設定すると入力ポートになります。

■ ＩＯ０ポート機能設定レジスタ（ＩＯ０ＳＬ）

ＩＯ０ポート機能設定レジスタはＩＯ０ポートの端子機能を設定するレジスタです。ＩＯ０ポートはオプションのＮＡＮＤ型フラッシュメモリーを制御する端子です。

	b7	b6	b5	b4	b3	b2	b1	b0
IO0SL	FME							
初期値	0							
方　向	R/W							

本レジスタのＦＭＥビット（ビット７）を１にするとＩＯ０ポート方向設定レジスタのすべてのビットは無効になります。
ＩＯ００ポート（ＡＬＥ信号)、ＩＯ０１ポート（ＣＬＥ信号)、ＩＯ０３ポート（ＦMWE信号)、ＩＯ０４ポート（ＦMWE信号)、ＩＯ０６ポート（ＦMＣＳ信号）及びＩＯ０７ポートが出力となり、ＩＯ０２ポート及びＩＯ０５ポート（ＦＭＲＢ信号）が入力になります。ＦMWＥに割り当てられたポートはＩＯ０ポート方向設定レジスタに書き込むとＨ８マイコンのWR信号が出力される区間でＬｏを出力します。ＦＭＲＥに割り当てられたポートはＩＯ０ポート方向設定レジスタを読み込むとＨ８マイコンのＲD信号が出力される区間でＬｏを出力します。
ＡＬＥ、ＣＬＥに割り当てられたポート及びＩＯ０７ポートは通常の出力ポート、ＦＭＲＢに割り当てられたポート及びＩＯ０２ポートはは通常の入力ポートとして取り扱いします。

４.３ ＩＯ１ポート

■ ＩＯ１ポートデータレジスタ（ＩＯ１Ｄ）

ＩＯ１ポートデータレジスタはＩＯ１０からＩＯ１７端子の状態をモニターしたり、端子の状態を設定するレジスタです。後述のＩＯ１ＤＩ方向設定レジスタまたはＩＯ１ＳＬ機能設定レジスタにより入出力の方向が決定します。

	B7	b6	b5	b4	b3	b2	b1	b0
IO1D	IO17	IO16	IO15	IO14	IO13	IO12	IO11	IO10
初期値	-	-	-	-	-	-	-	-
方　向	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ポートの設定が出力に設定されている場合、本レジスタのポートに対応するビットをセットする（１にする）とＨｉｇｈレベルを出力し、クリアする（0にする）とＬｏｗレベルを出力します。ポートの設定が入力に設定されている場合、本レジスタのポートに対応するビットを読み出すと、ポートに与えられた電位がＨｉｇｈの場合１が読み出され、Ｌｏｗの場合０が読み出されます。このレジスタはＩＯ１ポートを機能ポートとして利用する場合、書き込み無効となり、読み出した場合はバスの状態が読み出されます。

■ ＩＯ１ポート方向設定レジスタ（ＩＯ１ＤＩ）

ＩＯ１ポート方向設定レジスタはＩＯ１ポートの入出力方向を指定するレジスタです。１端子に付き１ビットが対応しています。このレジスタの設定はＩＯ１ポート機能設定レジスタのＢＵＳビットが１の場合無効になります。

	B7	b6	b5	b4	b3	b2	b1	B0
IO1DI	IOD17	IOD16	IOD15	IOD14	IOD13	IOD12	IOD11	IOD10
初期値	0	0	0	0	0	0	0	0
方　向	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

本レジスタのポートに対応するビットを１に設定するとそのポートは出力ポートになり、０を設定すると入力ポートになります。

■ ＩＯ１ポート機能設定レジスタ（ＩＯ１ＳＬ）

ＩＯ１ポート機能設定レジスタはＩＯ１ポートの端子機能を設定するレジスタです。ＩＯ１ポートはＣＰＵのバスとは独立した８ビットバスとして利用できます。

	b7	b6	b5	b4	b3	b2	b1	b0
IO1SL	BUS					SL2	SL1	SL0
初期値	0					0	0	0
方　向	R/W					R/W	R/W	R/W

本レジスタのＢＵＳビット（ビット７）を１に設定すると、ＩＯ１０からＩＯ１７がデータバスとして設定されます。また、ＳＬ２からＳＬ０ビットで、バスとして利用するアドレス範囲を選択できます。アドレス範囲として、６つの外部デバイス選択信号出力から選択できます。

選択アドレス	SL2	SL1	SL0
$\overline{\text{IOE0}}$ の出力範囲	0	0	0
$\overline{\text{IOE1}}$ の出力範囲	0	0	1
$\overline{\text{IOE2}}$ の出力範囲	0	1	0
$\overline{\text{IOE3}}$ の出力範囲	0	1	1
$\overline{\text{IOE4}}$ の出力範囲	1	0	0
$\overline{\text{IOE5}}$ の出力範囲	1	0	1

４．４ ＩＯ３ポート

■ ＩＯ３ポートデータレジスタ（ＩＯ３Ｄ）

ＩＯ３ポートデータレジスタはＩＯ３０からＩＯ３５端子の状態をモニターしたり、端子の状態を設定するレジスタです。ＩＯ３ＤＩ方向設定レジスタまたはＩＯ３ＳＬ機能設定レジスタにより入出力の方向が決定します。

	b7	b6	b5	b4	b3	b2	b1	b0
IO3D			IO35	IO34	IO33	IO32	IO31	IO30
初期値			-	-	-	-	-	-
方　向			R/W	R/W	R/W	R/W	R/W	R/W

ポートの設定が出力に設定されている場合、本レジスタのポートに対応するビットをセットする（1にする）とＨｉｇｈレベルを出力し、クリアする（0にする）とＬｏｗレベルを出力します。ポートの設定が入力に設定されている場合、本レジスタのポートに対応するビットを読み出すと、ポートに与えられた電位がＨｉｇｈの場合１が読み出され、Ｌｏｗの場合０が読み出されます。

■ ＩＯ３ポート方向設定レジスタ（ＩＯ３ＤＩ）

ＩＯ３ポート方向設定レジスタはＩＯ３ポートの入出力方向を指定するレジスタです。１端子に付き１ビットが対応しています。ＩＯ３ポート機能設定レジスタのポートに対応するビットが１の場合、そのポートに対応する本レジスタのビットは無効になります。

	b7	B6	b5	b4	b3	b2	b1	b0
IO3DI			IOD35	IOD34	IOD33	IOD32	IOD31	IOD30
初期値			1	0	0	0	0	0
方　向			R/W	R/W	R/W	R/W	R/W	R/W

本レジスタのポートに対応するビットを１に設定するとそのポートは出力ポートになり、０を設定すると入力ポートになります。

■ ＩＯ３ポート機能設定レジスタ（ＩＯ３ＳＬ）

ＩＯ３ポート機能設定レジスタはＩＯ３ポートの端子機能を設定するレジスタです。ＩＯ３ポートは３章で説明した外部デバイス選択信号を出力するポートになっています。

	b7	b6	b5	b4	b3	b2	b1	b0
IO3SL		IOE6E	IOE5E	IOE4E	IOE3E	IOE2E	IOE1E	IOE0E
初期値		1	1	0	0	0	0	0
方　向		R/W	R/W	R/W	R/W	R/W	R/W	R/W

本レジスタのポートに対応する端子のビットを１にすると、ＩＯ３ＤＩレジスタの設定に関わらず、外部デバイス選択信号出力端子になります。IO6E ﾋﾞｯﾄを１にセットすると、本モジュールで使用していないアドレス空間のアドレスデコード信号をＩＯ３５ポートより出力するようになっております。未使用アドレスを利用する場合はこのﾋﾞｯﾄを ON にし、必ずこのﾎﾟｰﾄをアドレスデコード信号としてご利用ください。モードごとに未使用の外部アドレスはさまざまですが、システムモードにおいて使用されるアドレスもあります。この信号を使用することによりアドレスの衝突がないようにしてください。

４．５ ウォッチドックタイマレジスタ（ＷＤＴＣ）

ウォッチドックタイマーレジスタはオプションの RTC を搭載した時に有効となります。
ＲＴＣのＴＭＯＵＴ端子より周期的に出力されるパルスを使用し、ウォッチドックタイマーとして機能させます。本レジスタを周期的に読み出すことにより、リセットを解除します。

	b7	b6	b5	b4	b3	b2	b1	B0
WDTC	RST							WDTE
初期値	0							0
方　向	R/W							R/W

■ RST ビット（ビット７）

本ビットはウォッチドックタイマーによりリセットが掛かった場合 1 がセットされます。電源投入時のリセット以外は０にクリアされませんので、ウォッチドックタイマーによるリセットを判断できます。

■ WDTE ビット（ビット 0）

本ビットに１を書き込むとウォッチドックタイマーとして機能します。

４．６ システム制御レジスタ（ＳＹＳＣＴＬ）

システム制御レジスタは**ＡＴ８０１０**のモードを確認したり、周辺機器へのリセットを出力するレジスタです。

	b7	b6	b5	b4	b3	b2	b1	b0
SYSCTL	PRES		RES	HAE	MD3	MD2	MD1	MD0
初期値	0		0	0	x	x	x	x
方　向	R/W	R/W	R/W	R/W	R	R	R	R

■ ＰＲＥＳビット（ビット７）

本ビットをセットすると～ＰＲＥＳ端子がＬｏに、クリアするとＨｉになります。

■ ＲＥＳビット（ビット５）

本ビットに１を書き込むとシステム全体がリセットします。Ｈ８のシステムクロックで３２クロックの区間、システム全体がリセット状態になります。

■ ＨＡＥビット（ビット４）

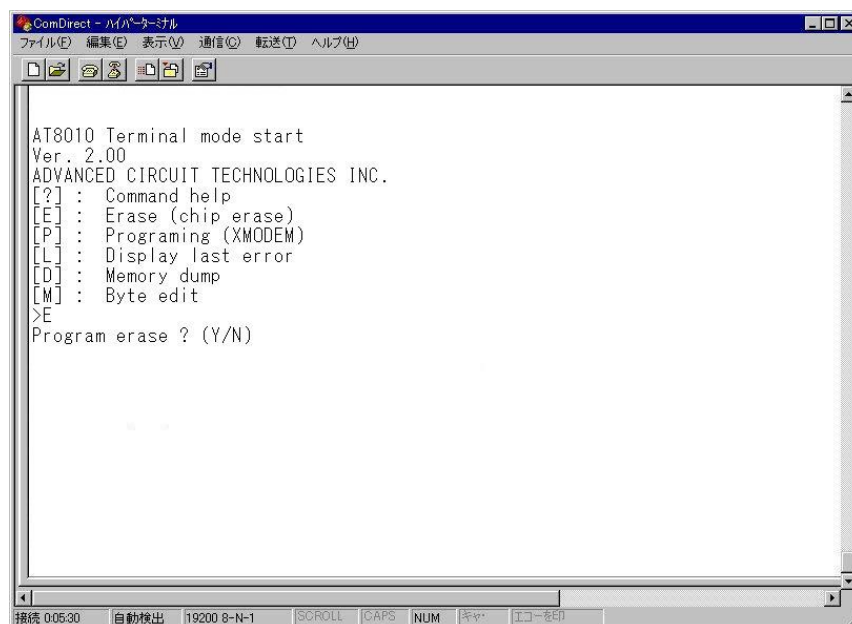
本ビットをセットすると～ＰＲＥＳ端子がＬｏに、クリアするとＨｉになります。ＭＤ３－ＭＤ０ビット（ビット３－ビット０）モード端子の状態をモニターできます。

第５章　プログラムの書き換え

ＡＴ８０１０はＨ８マイコンのＳＣＩのチャンネル０を利用し、プログラムの変更ができるようになっています（XＭｏｄｅｍ（チェックサム）プロトコルが利用できるターミナルソフトをご利用ください)。**ＡＴ８０１０**のＴＸＤ０、ＲＸＤ０端子をＲＳ２３２Ｃレベルコンバーターを介してパソコンのＲＳ２３２Ｃコネクタに接続します。パソコンのターミナルソフトを起動し、19200bps、ストップビット１、パリティーなし、データ長８ビットに設定します。

モードをシステムモード０になるようにモード端子を設定し、電源をＯＮにしてください。

上記の手続きを行うと次画面の様にメニューが表示されます。

画面Ａ

５．１ コマンドの一覧

プログラムの書き換えを説明する前にメニューに表示されているコマンド一覧を記載します。

	コマンド	名　　称	機　　能
1	?	Command help	コマンドの一覧を表示します。
2	E	Erase (chip erase)	メモリー消去
3	P	Programing (XMODEM)	プログラムの書き換え
4	L	Display last error	エラー発生時の最後のエラー表示
5	D	Memory Dump	メモリーダンプ
6	M	Byte edit	メモリー内容の書き換え

５．２ コマンド説明

５.２.１【？】: Command help

■ **説明**

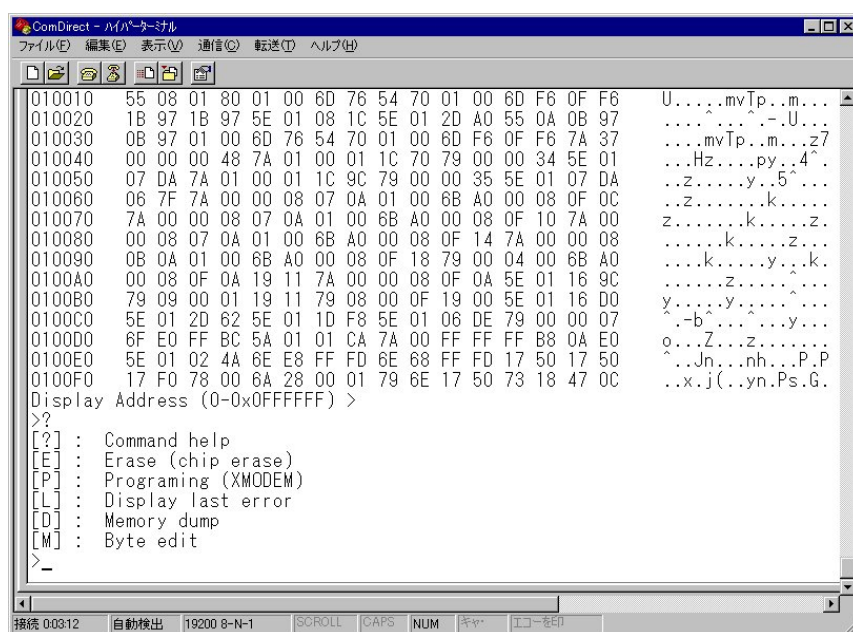
ここで使用できるコマンド一覧を表示します。

■ **フォーマット**

？

■ **例**

“？”キーを押下するとコマンド一覧が表示されます。(画面Ｂ)

画面Ｂ

５.２.２【E】: Erase (chip erase)

■ **説明**

プログラムを書き込むメモリの消去を行います。
プログラムを書き換える時、一度メモリを消去してください。

■ **フォーマット**

E

■ **例**

>E
Program erase ? (Y/N)
Erase done
>

“E”キーを押下すると“Ｐｒｏｇｒａｍ ｅｒａｓｅ？（Ｙ／Ｎ)”メッセージが表示されます。“Y”キーを押下するとメモリ消去を開始し、消去終了すると
“Ｅｒａｓｅ ｄｏｎｅ”メッセージ表示で消去が正常に終了します。
“N”キーを押下するとメモリは消去されず次のコマンド待ち状態になります。

５.２.３【Ｐ】: Programing (XMODEM)

■ **説明**

プログラムを書き換えを行います。

■ **フォーマット**

Ｐ

■ **例**

５．３項のハイパーターミナルによるプログラム書き換え例を参照してください。

５.２.４【Ｌ】: Display last error

■ **説明**

エラー発生時の最後のエラー状態を表示します。

■ **フォーマット**

Ｌ

■ **例**

５.２.５【Ｄ】:　Memory Dump

■ 説明

指定されたメモリアドレスの内容を表示します。

■ **フォーマット**

D

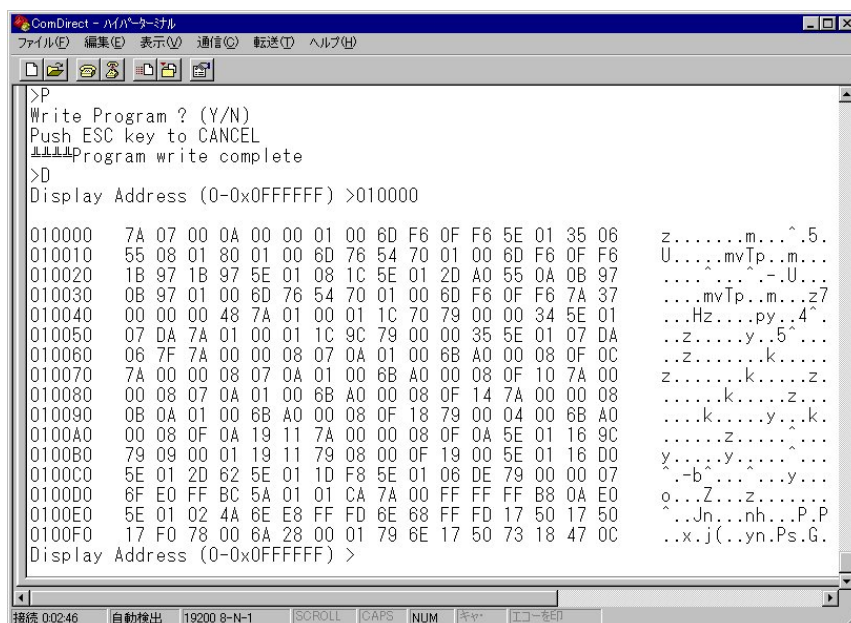
＜Address＞

■ **例**

```
>D
Display Address (0-0x0FFFFFF) >010000

010000   7A 07 00 0A 00 00 01 00 6D F6 0F F6 5E 01 35 06   z.......m...^.5.
010010
   ・
   ・
   ・
Display Address (0-0x0FFFFFF) >
```

“D”キーを押下すると
“Ｄｉｓｐｌａｙ　Ａｄｄｒｅｓｓ（０－０ｘＦＦＦＦＦＦ）＞”メッセージが表示さ、表示するメモリ内容のアドレスの入力待ちになります。アドレスの入力は０～ＦＦＦＦＦＦのＨＥＸ（１６進）コードで指定します。ここでは“０１００００”と入力します。するとメモリアドレスの０１００００番地～０１００ＦＦ番地（ＨＥＸ）の２５６バイトの内容が表示されます。(画面Ｃ)
２５６バイト表示後“Ｄｉｓｐｌａｙ　Ａｄｄｒｅｓｓ（０－０ｘＦＦＦＦＦＦ）＞” メッセージが表示され入力待ちになります。
“Ｅｎｔｅｒ”キーを押下すると次のアドレスのメモリ内容２５６バイトが表示されます。
終了するときは“ＥＳＣ”キーを押下してください。

画面Ｃ

５.２.６【Ｍ】: Byte edit

■ 説明

指定されたメモリアドレスの内容を１バイト単位で書き換えをおこないます。

■ フォーマット

Ｍ

＜Address＞

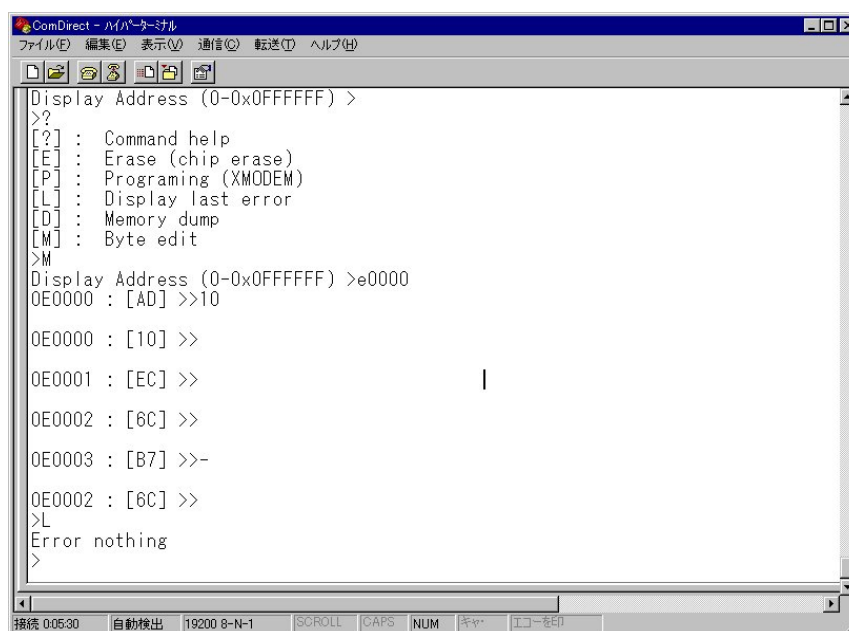
■ 例

```
>M
Display Address (0-0x0FFFFFF) >e0000
0E0000 : [AD] >>10

0E0001 : [EC] >>
```

“Ｍ”キーを押下すると

“Ｄｉｓｐｌａｙ Ａｄｄｒｅｓｓ（０－０ｘＦＦＦＦＦＦ）＞”メッセージが表示さ、書き換えるメモリのアドレスの入力待ちになります。アドレスの入力は０～ＦＦＦＦＦＦのＨＥＸ（１６進）コードで指定します。ここでは“ｅ００００”と入力します。“０Ｅ００００：［ＡＤ］＞＞”と０Ｅ００００番地の内容“ＡＤ”がＨＥＸコードで表示され、次に変更するデータの入力待ちになります。この例ではＨＥＸコードの“１０”と入力し、“Ｅｎｔｅｒ”キーを押下します。(画面Ｄ)

画面Ｄ

“Ｅｎｔｅｒ”キー：アドレスが＋１され、入力待ち。

“＾”キー　　　：アドレスは変わらず、入力待ち。

“－”キー　　　：アドレスが－１され、入力待ち。

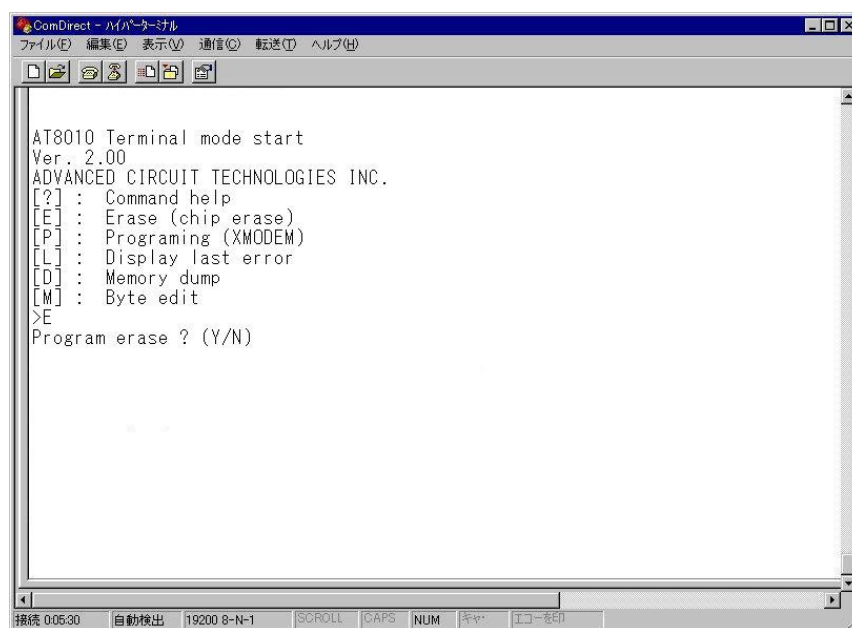
“ＥＳＣ”キー　：メモリ書き換え終了し、コマンド入力待ち。

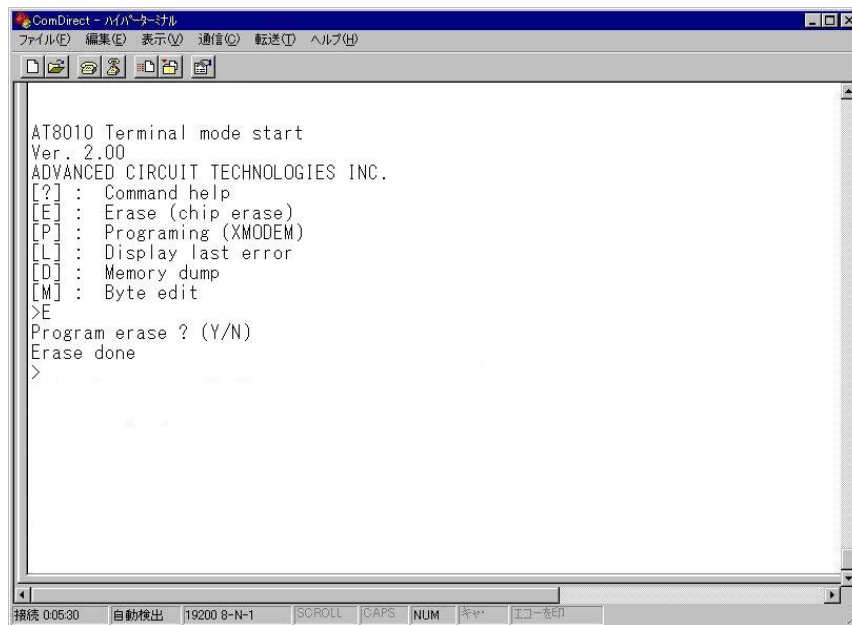
５．３ ハイパーターミナルによるプログラム書き換え例

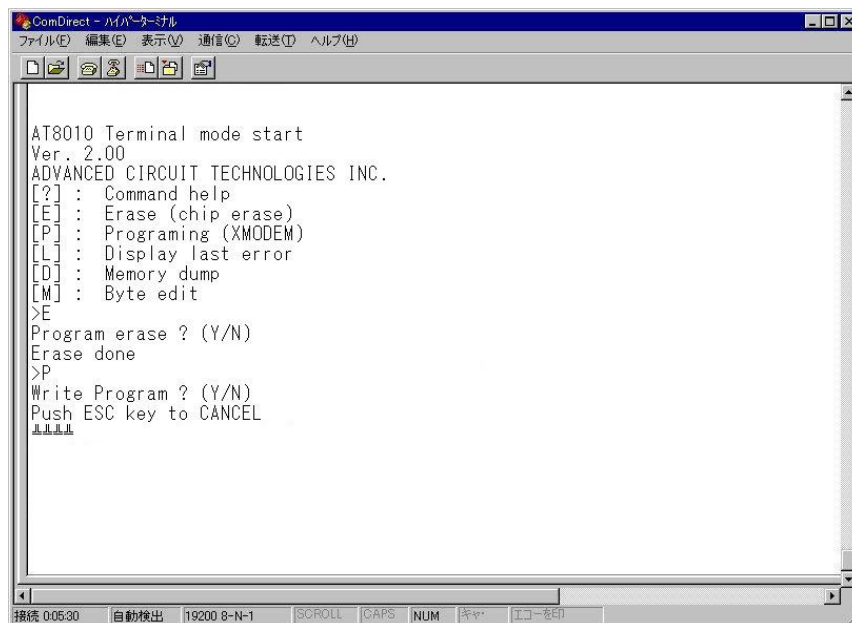
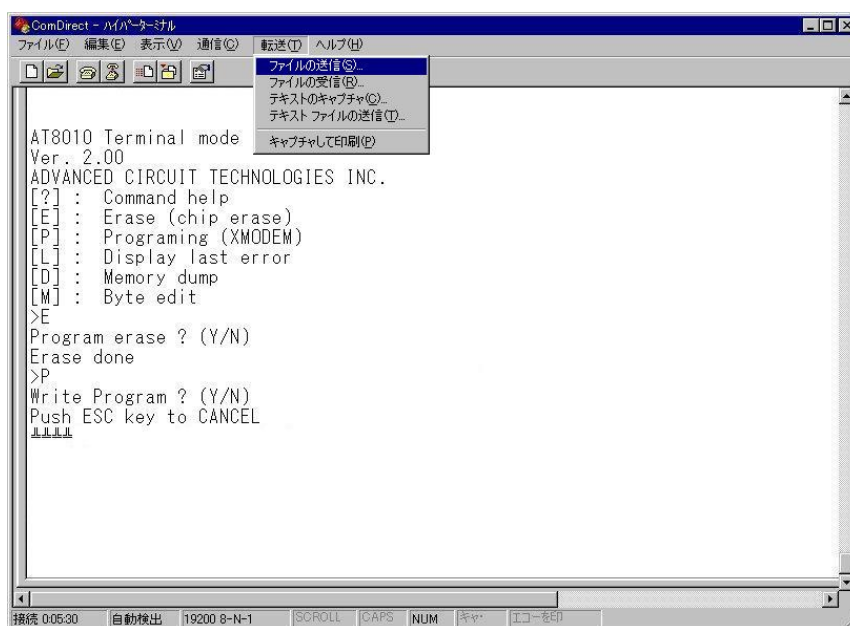
パソコンのターミナルソフトを起動し、19200bps、ストップビット 1、パリティーなし、データ長８ビットに設定します。(ハイパーターミナルがない場合は Windows95、98、ME、2000 にありますのでインストールしてください。)

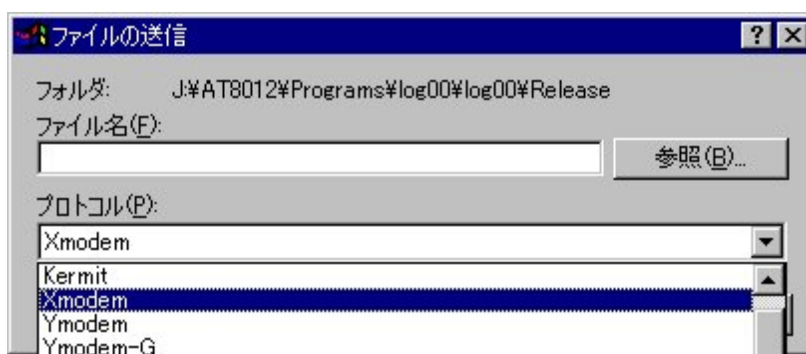
モードをシステムモード０になるようにモード端子を設定し、電源をＯＮにしてください。

上記の手続きを行うと画面Ｅの様にメニューが表示されます。

画面Ｅ

プログラムを書き換える場合まず、メモリーを消去する必要がありますので、メニューの“E”コマンドを入力し、“Y”キーを入力するとメモリーが消去されます。(画面 F)

画面Ｆ

次に、“P”コマンドを入力し、“Y”を入力するとプログラムの転送待ち状態になりますので、ハイパーターミナルのプログラムの転送を行います。(画面ＧからＫ)

画面Ｇ

画面Ｈ

ハイパーターミナルのファイル転送を選択

画面Ｉ

プロトコルをＸｍｏｄｅｍに設定し、転送するファイル名称を入力する。
転送するファイルはモトローラＨＥＸフォーマットファイルを指定してください。

画面Ｊ

Xmodem ファイル送信 - ComDirect

送信中: J:¥AT8012¥Programs¥log00¥log00¥Release¥log00.mot

パケット: 32　エラー チェック: Checksum

再試行: 0　再試行の回数: 0

最新のエラー:

ファイル:　4K / 93K

経過時間: 00:00:03　残り: 00:01:13　スループット: 1237 cps

キャンセル　cps/bps(C)

データ転送中の画面で、パケットの数値がしばらくたっても上がらない場合はキャンセルを押してから、ＥＳＣキーを入力してください。
その後、“Ｌ”コマンドを入力し、エラーが発生していないかチェックしてください。

画面Ｋ

正常に終了すると上記画面になります。

第６章　ＲＴＣの使い方

オプションで搭載されるリアルタイムクロック（ＲＴＣ）は株式会社リコー社製「ＲＳ5Ｃ317Ａ」です。このデバイスはシリアルインターフェースですが、ＡＴ８０１０に搭載されたＦＰＧＡでパラレル⟷シリアル変換をおこなっていますので、プログラミングの際はレジスタにアドレス及びデータを書き込むだけで設定可能です。ＲＴＣ設定レジスタを以下に記します。

６．１ レジスター覧

レジスタ名	アドレス	機　　　　能
RTSTS	0xFFF610	ステータス・データレジスタ
RTCR	0xFFF611	読み込みアドレス設定レジスタ
RTCW	0xFFF612	書き込みアドレス・データ設定レジスタ

６．２ レジスタ機能

■ ステータス・データレジスタ

	b7	B6	b5	b4	b3	b2	b1	b0
RTSTS	BSY				D3	D2	D1	D0
初期値	-				-	-	-	-
方　向	R(/W)	(W)	(W)	(W)	R(/W)	R(/W)	R(/W)	R(/W)

本レジスタはRTCへの送受信ステータス読み込み、受信データの格納、RTCW、RTCRレジスタで設定された内容の送信を開始するためのレジスタです。書き込みを行うと、その直前に設定したレジスタ（RTCRまたはRTCWレジスタ）の内容をRTCに転送（受信）します。読み込みを行うとRTCにデータを転送中はBSYビットが1になります。BSYビットが1の時は本レジスタ及びRTCR、RTCWへのライトはしないでください。RTCRレジスタ書き込み後、本レジスタを書き込み、BSYビットが0になったとき、RTCからのデータ受信終了となり、D3からD0ビットにRTCのデータがセットされます。

■ 読み込みアドレス設定レジスタ

	b7	B6	b5	b4	b3	b2	b1	b0
RTCR	A3	A2	A1	A0				
初期値	-	-	-	-				
方　向	W	W	W	W				

本レジスタはRTCの読み出しアドレスを設定するレジスタです。A3からA0ビットにRTCの読み出しアドレスを設定し、RTSTSレジスタをライトしてください。RTSTSレジスタのBSYビットが1から0になったとき、指定されたアドレスのデータがRTSTSレジスタのD3からD0ビットにセットされます。

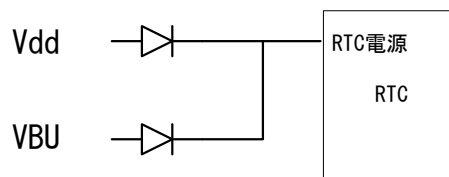
■ 書き込みアドレス・データ設定レジスタ

	b7	B6	b5	b4	b3	b2	b1	b0
RTCW	A3	A2	A1	A0	D3	D2	D1	D0
初期値	-	-	-	-	-	-	-	-
方　向	W	W	W	W	W	W	W	W

本レジスタは RTC への書き込みアドレス及びデータを設定するレジスタです。A3 から A0 ビットに RTC へ書き込むアドレス及びデータを設定し、RTSTS レジスタをライトしてください。RTSTS レジスタの BSY ビットが 1 から 0 になったとき、指定されたアドレスにデータがセットされます。

６．３　ＲＴＣ電源回路

AT8010 は RTC のカレンダーを保持するためのバックアップ用電源端子が用意されています。RTC をバックアップする場合は VBU 端子を利用してください。

■ RTC 電源回路概要

６．４ プログラム例

ＲTC のカレンダー・時計を設定、読み出しをするプログラム例はを記述します。
各レジスタ詳細については株式会社リコーのアラーム機能付き高機能リアルタイムクロックＲＳ５Ｃ３１７Ａ／Ｂアプリケーションマニュアルを参照してください。

■ プログラム例

ここではＲＴＣの初期設定、時計・カレンダの設定および呼び出しのプログラム例を記述します。

```
/*****************************************************************************
   RTC プログラム例
*****************************************************************************/
typedef enum { FALSE = 0, TRUE } boolean;

/*  RTC レジスタアドレス定義          */
#define RTCW       (*(volatile unsigned char *)0xFFF612) /* 書込ｱﾄﾞﾚｽ･ﾃﾞｰﾀ設定 ﾚｼﾞｽﾀ */
#define RTCR       (*(volatile unsigned char *)0xFFF611) /* 読込ｱﾄﾞﾚｽ設定 ﾚｼﾞｽﾀ      */
#define RTSTS      (*(volatile unsigned char *)0xFFF610) /* ｽﾃｰﾀｽ･ﾃﾞｰﾀ ﾚｼﾞｽﾀ         */

/*  領域定義                 */
const unsigned char rtcreg[14] = {    /* 時計・ｶﾚﾝﾀﾞ用ｶｳﾝﾀﾚｼﾞｽﾀのｱﾄﾞﾚｽ         */
                  0x00, 0x10,         /* [0]：時刻１秒桁ﾚｼﾞｽﾀ , [1]：時刻 10 秒桁ﾚｼﾞｽﾀ */
                  0x20, 0x30,         /* [2]：時刻１分桁ﾚｼﾞｽﾀ , [3]：時刻 10 分桁ﾚｼﾞｽﾀ */
                  0x40, 0x50,         /* [4]：時刻１時桁ﾚｼﾞｽﾀ , [5]：時刻 10 時桁ﾚｼﾞｽﾀ */
                  0x60,               /* [6]：曜日ｶｳﾝﾀﾚｼﾞｽﾀ                          */
                  0x80, 0x90,         /* [7]：ｶﾚﾝﾀﾞ 1 日桁ﾚｼﾞｽﾀ , [8]：ｶﾚﾝﾀﾞ 10 日桁ﾚｼﾞｽﾀ */
                  0xA0, 0xB0,         /* [9]：ｶﾚﾝﾀﾞ 1 月桁ﾚｼﾞｽﾀ , [10]：ｶﾚﾝﾀﾞ 10 月桁ﾚｼﾞｽﾀ */
                  0xC0, 0xD0,         /* [11]：ｶﾚﾝﾀﾞ 1 年桁ﾚｼﾞｽﾀ , [12]：ｶﾚﾝﾀﾞ 10 年桁ﾚｼﾞｽﾀ */
                  0x00 } ;

                            /* 時計・ｶﾚﾝﾀﾞの設定ﾃﾞｰﾀ 2000 年 6 月 26 日 0 時 0 分 0 秒    */
const unsigned char setdate[13] = {
                  0x00, 0x01,         /* [0]：10 年桁 00 , [1]：1 年桁 01            */
                  0x00, 0x06,         /* [2]：10 月桁 00 , [3]：1 月桁 06            */
                  0x02, 0x06,         /* [4]：10 日桁 02 , [5]：1 日桁 06            */
                  0x02, 0x00,         /* [6]：曜日桁 02 , [7]：10 時桁 00            */
                  0x00, 0x00,         /* [8]：1 時桁 00  , [9]：10 分桁 00           */
                  0x00, 0x00,         /* [10]：1 分桁 00  , [11]：10 秒桁 00         */
                  0x00 } ;            /* [11]：1 秒桁 00                             */

unsigned char data[14] ;              /* 時計・ｶﾚﾝﾀﾞ読込ﾃﾞｰﾀ                         */

/*  関数プロトタイプ宣言                */
void main ( void ) ;
boolean RtcInit( void ) ;
void RtcSetDateTime( unsigned char    *dt ) ;
void RtcGetDateTime( unsigned char    *dt ) ;
```

```
/*-----------------------------------------------------------------------------------
    メインルーチン
-----------------------------------------------------------------------------------*/
void main(void)
{
  RtcInit() ;                          /* RTC 初期化処理           */
  RtcSetDateTime( setdata ) ;          /* カレンダーと時刻設定     */
  while(1){
         RtcGetDateTime( data ) ;      /* 日付と時刻呼び出し       */
  }
}

/*-----------------------------------------------------------------------------------
    RTC 初期化 発振停止が検出（制御レジスタ 1 の XSTP=0）されない時エラー（FALSE）終了。
-----------------------------------------------------------------------------------*/
boolean RtcInit( void )
{
short i ;
    /*
      読込ｱﾄﾞﾚｽ設定ﾚｼﾞｽﾀ(RTCR)に制御ﾚｼﾞｽﾀ 1 ｱﾄﾞﾚｽ(0xE0)をｾｯﾄし、ｽﾃｰﾀｽ･ﾃﾞｰﾀﾚｼﾞｽﾀ(RTSTS)
      に 0x00 を書き込む。その後、RTSTS ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄが 1 から 0 になったとき、RTSTS ﾚｼﾞｽﾀ
      の D3～D0 ﾋﾞｯﾄﾃﾞｰﾀ（制御ﾚｼﾞｽﾀ 1 のﾃﾞｰﾀ）を読む。
      制御ﾚｼﾞｽﾀ 1 ﾃﾞｰﾀの XSTP ﾋﾞｯﾄが OFF(=0)のときエラーで処理を終了する。
      XSTP ﾋﾞｯﾄが ON(=1)のとき発振停止検出時で次の処理へ。
    */
    RTCR = 0xE0 ;
    RTSTS = 0x00 ;
    while( RTSTS & 0x80 ) ;
    if( ( RTSTS & 0x02 ) == 0)
        return FALSE ;

    /*
      1.割込周期ﾚｼﾞｽﾀのクリア。
        書込ｱﾄﾞﾚｽ･ﾃﾞｰﾀ設定 ﾚｼﾞｽﾀ(RTCW)に割込周期ﾚｼﾞｽﾀのｱﾄﾞﾚｽ(0x70)とｸﾘｱﾃﾞｰﾀ(0x00)ｾｯﾄ。
      2.バンク 1 設定。
        RTCW ﾚｼﾞｽﾀに制御ﾚｼﾞｽﾀ 2 のｱﾄﾞﾚｽ(0xF0)とﾃﾞｰﾀ(0x0B)をｾｯﾄ。
        ﾃﾞｰﾀ D3 ﾋﾞｯﾄ：時間表示選択ﾋﾞｯﾄ(ON:24 時間制)
             D2 ﾋﾞｯﾄ：ﾀｲﾏｰ用ｶｳﾝﾀのﾘｾｯﾄﾋﾞｯﾄ(OFF:変化なし)
             D1 ﾋﾞｯﾄ：ﾊﾞﾝｸ切替ﾋﾞｯﾄ(ON:ｱﾗｰﾑﾚｼﾞｽﾀを選択)
             D0 ﾋﾞｯﾄ：テスト用ﾋﾞｯﾄ(ON:通常動作ﾓｰﾄﾞ)
      3.ALE ﾋﾞｯﾄ ｸﾘｱ
        RTCW ﾚｼﾞｽﾀにｱﾗｰﾑ 10 時ﾚｼﾞｽﾀのｱﾄﾞﾚｽ(0x50)とﾃﾞｰﾀ(0x00)をｾｯﾄ。
        ﾃﾞｰﾀ ALE ﾋﾞｯﾄ OFF(ｱﾗｰﾑ一致動作無効)
      4.ADJ ﾋﾞｯﾄ 設定
        RTCW ﾚｼﾞｽﾀに制御ﾚｼﾞｽﾀ 1 のｱﾄﾞﾚｽ(0xE0)とﾃﾞｰﾀ(0x03)をｾｯﾄ。
        ﾃﾞｰﾀ WTEN ﾋﾞｯﾄ：時刻ｶｳﾝﾄ動作の設定ﾋﾞｯﾄ(ON:時計ｶｳﾝﾀの 1 秒桁上げ有効)
             ADJ ﾋﾞｯﾄ：±30 秒ｱｼﾞｬｽﾄﾋﾞｯﾄ(ON:秒桁合わせ)
        RTCW ﾚｼﾞｽﾀに出力したあと、RTSTS ﾚｼﾞｽﾀに 0x00 を書き込む。その後、
        RTSTS ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄが 1 から 0 になったことを確認。
    */
```

```
RTCW = 0x70 | 0x00 ;             /* 割込周期ﾚｼﾞｽﾀ ｸﾘｱ         */
RTSTS = 0 ;
while( RTSTS & 0x80 ) ;
RTCW  = 0xF0 | 0x0B ;            /* ﾊﾞﾝｸ 1 設定               */
RTSTS = 0 ;
while( RTSTS & 0x80 ) ;
RTCW  = 0x50 | 0x00 ;            /* ALE ﾋﾞｯﾄ ｸﾘｱ              */
RTSTS = 0 ;
while( RTSTS & 0x80 ) ;
RTCW  = 0xE0 | 0x03 ;            /* ADJ ﾋﾞｯﾄ設定              */
RTSTS = 0 ;
while( RTSTS & 0x80 ) ;

/*
  RTCR ﾚｼﾞｽﾀに制御ﾚｼﾞｽﾀ 1 ｱﾄﾞﾚｽ(0xE0)をｾｯﾄし、ｽﾃｰﾀｽ･ﾃﾞｰﾀﾚｼﾞｽﾀ(RTSTS)に 0x00 を書き込む。
  その後、RTSTS ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄが 1 から 0 になったとき、RTSTS ﾚｼﾞｽﾀの D3～D0 ﾋﾞｯﾄﾃﾞｰﾀ
  (制御ﾚｼﾞｽﾀ 1 のﾃﾞｰﾀ) を読む。制御ﾚｼﾞｽﾀ 1 ﾃﾞｰﾀの BSY ﾋﾞｯﾄが OFF(=0)になったら次の処理へ。
  BSY ﾋﾞｯﾄが OFF になるまで上記を繰り返す。
*/
for( ; ; )
{
  RTCR  = 0xE0 ;
  RTSTS = 0x00 ;
  while( RTSTS & 0x80 ) ;
  if( !( RTSTS & 0x01 ) )
    break ;
  }

/*
  1.RTCW ﾚｼﾞｽﾀに制御ﾚｼﾞｽﾀ 1 のｱﾄﾞﾚｽ(0xE0)とﾃﾞｰﾀ(0x00)を書き込む。
    ﾃﾞｰﾀ WTEN ﾋﾞｯﾄ : 時刻ｶｳﾝﾄ動作の設定ﾋﾞｯﾄ(OFF:時計ｶｳﾝﾀの 1 秒桁上げ停止)
         ADJ ﾋﾞｯﾄ : ±30 秒ｱｼﾞｬｽﾄﾋﾞｯﾄ(OFF:通常動作)
  2.RTCW ﾚｼﾞｽﾀに制御ﾚｼﾞｽﾀ 2 のｱﾄﾞﾚｽ(0xF0)とﾃﾞｰﾀ(0x09)を書き込む。
    ﾃﾞｰﾀ D3 ﾋﾞｯﾄ : 時間表示選択ﾋﾞｯﾄ(ON:24 時間制)
         D2 ﾋﾞｯﾄ : ﾀｲﾏｰ用ｶｳﾝﾀのﾘｾｯﾄﾋﾞｯﾄ(OFF:変化なし)
         D1 ﾋﾞｯﾄ : ﾊﾞﾝｸ切替ﾋﾞｯﾄ(OFF:時計・ｶﾚﾝﾀﾞ用ｶｳﾝﾀを選択)
         D0 ﾋﾞｯﾄ : テスト用ﾋﾞｯﾄ(ON:通常動作ﾓｰﾄﾞ)
  RTCW ﾚｼﾞｽﾀに出力したあと、次の動作を行う。
  RTSTS ﾚｼﾞｽﾀに 0x00 を書き込む。その後、RTSTS ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄが 1 から 0 になったことを確認。
*/
RTCW  = 0xE0 | 0x00 ;
RTSTS = 0 ;
while( RTSTS & 0x80 ) ;
RTCW  = 0xF0 | 0x09 ;            /* 24 時間制設定             */
RTSTS = 0 ;
while( RTSTS & 0x80 ) ;
```

```
    /*
      時計・ｶﾚﾝﾀﾞ用ｶｳﾝﾀのﾚｼﾞｽﾀのｸﾘｱ
    */
    for( i=0; i<13; i++ )
    {
      RTCW = rtcreg[i] ;              /* RTC ｶｳﾝﾀ ｸﾘｱ    */
      RTSTS = 0 ;
      while( RTSTS & 0x80 ) ;
    }

    return TRUE ;
}

/*-----------------------------------------------------------------
         カレンダと時刻設定
-----------------------------------------------------------------*/
void RtcSetDateTime( unsigned char   *dt )
{
short i ;

    /*
      RTCW ﾚｼﾞｽﾀに制御ﾚｼﾞｽﾀ 1 のｱﾄﾞﾚｽ(0xE0)とﾃﾞｰﾀ(0x01)を書き込む。
      ﾃﾞｰﾀ WTEN ﾋﾞｯﾄ : 時刻ｶｳﾝﾄ動作の設定ﾋﾞｯﾄ(OFF:時計ｶｳﾝﾀの 1 秒桁上げ停止)
           ADJ ﾋﾞｯﾄ : ±30 秒ｱｼﾞｬｽﾄﾋﾞｯﾄ(ON:秒桁合わせ)
      RTSTS ﾚｼﾞｽﾀに 0x00 を書き込む。その後、RTSTS ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄが 1 から 0 になったことを確認。
    */
    RTCW  = 0xE0 | 0x01 ;            /* ADJ セット      */
    RTSTS = 0 ;
    while( RTSTS & 0x80 ) ;

    /*
      RTCR ﾚｼﾞｽﾀに制御ﾚｼﾞｽﾀ 1 ｱﾄﾞﾚｽ(0xE0)をｾｯﾄし、ｽﾃｰﾀｽ･ﾃﾞｰﾀﾚｼﾞｽﾀ(RTSTS)に 0x00 を書き込む。
      その後、RTSTS ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄが 1 から 0 になったとき、RTSTS ﾚｼﾞｽﾀの D3～D0 ﾋﾞｯﾄﾃﾞｰﾀ
      (制御ﾚｼﾞｽﾀ 1 のﾃﾞｰﾀ) を読む。制御ﾚｼﾞｽﾀ 1 ﾃﾞｰﾀの BSY ﾋﾞｯﾄが OFF(=0)になったら次の処理へ。
      BSY ﾋﾞｯﾄが OFF になるまで上記を繰り返す。
    */
    for( ; ; )
    {
      RTCR  = 0xE0 ;
      RTSTS = 0x00 ;
      while( RTSTS & 0x80 ) ;
      if( !( RTSTS & 0x01 ) )
        break ;
    }

    /*
      時計・ｶﾚﾝﾀﾞ用ｶｳﾝﾀのﾚｼﾞｽﾀにデータを設定する。
    */
```

```
    for( i=0; i<13; i++ )
    {
      RTCW  = rtcreg[i] | setdate[12-i] ;
      RTSTS = 0x00 ;
      while( RTSTS & 0x80 ) ;
    }
}

/*---------------------------------------------------------------
         日付と時刻呼び出し
---------------------------------------------------------------*/
void RtcGetDateTime( unsigned char   *dt )
{
short i ;

    /*
      RTCR ﾚｼﾞｽﾀに制御ﾚｼﾞｽﾀ 1 ｱﾄﾞﾚｽ(0xE0)をｾｯﾄし、ｽﾃｰﾀｽ･ﾃﾞｰﾀﾚｼﾞｽﾀ(RTSTS)に 0x00 を書き込む。
      その後、RTSTS ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄが 1 から 0 になったとき、RTSTS ﾚｼﾞｽﾀの D3～D0 ﾋﾞｯﾄﾃﾞｰﾀ
      (制御ﾚｼﾞｽﾀ 1 のﾃﾞｰﾀ) を読む。制御ﾚｼﾞｽﾀ 1 ﾃﾞｰﾀの BSY ﾋﾞｯﾄが OFF(=0)になったら次の処理へ。
      BSY ﾋﾞｯﾄが OFF になるまで上記を繰り返す。
    */
    for( ; ; )
    {
      RTCR  = 0xE0 ;
      RTSTS = 0x00 ;
      while( RTSTS & 0x80 ) ;
      if( !( RTSTS & 0x01 ) )
        break ;
    }

    /*
      時計・ｶﾚﾝﾀﾞ用ｶｳﾝﾀのﾚｼﾞｽﾀからデータを読み込む。
    */
    for( i=0; i<13; i++ )
    {
      RTCR  = rtcreg[i] ;
      RTSTS = 0x00 ;
      while( RTSTS & 0x80 ) ;
      *(dt+12-i) = RTSTS & 0x7F ;
    }
}
```

第７章　ＮＡＮＤフラッシュメモリの使い方

ＮＡＮＤ型フラッシュメモリーは株式会社東芝社製「ＴＣ5864ＦＴ」の搭載しております。このメモリーの使用方法は通常メモリーと異なり、アドレスバスがありません。読み出し、書き込みアドレスはコマンドで与えます。詳細は製品のマニュアルを参照してください。

ＡＴ８０１０はこのメモリーにアクセスするための専用ポートを用意しています。（ＩＯ０ポート）
ＩＯ０ポートの機能設定レジスタのＦＭＥビットに１をセットすると、フラッシュメモリーのアクセス専用ポートに設定されます。各ビットがフラッシュメモリーの制御端子に接続されておりソフトウェアによって制御端子をコントロールします。

また、アドレスＦＦＦ６０４Ｈはフラッシュメモリー専用のデータ読み出し／書き込みレジスタとなり、このレジスタに読み出し／書き込みを行うことにより、フラッシュメモリーのＲＤ／ＷＲ端子をアサートします。

７．１　プログラム例

ここではＮＡＮＤフラッシュメモリの初期設定、読み出し／書き込み、消去のプログラム例を記述します。
各レジスタ詳細については株式会社東芝６４メガビット(８Ｍ×８ビット) ＣＭＯＳ ＮＡＮＤ Ｅ²ＰＲＯＭ「ＴＣ５８６４ＦＴ」資料を参照してください。

■ プログラム例

```
/*******************************************************************************
   NAND フラッシュメモリ  プログラム例
*******************************************************************************/
#include <machine.h>                         /* ﾗｲﾌﾞﾗﾘ関数用ﾍｯﾀﾞﾌｧｲﾙ              */
#include "h83042s.h"                         /* 周辺ﾚｼﾞｽﾀ定義ﾍｯﾀﾞﾌｧｲﾙ             */

/*  構造体定義              */
typedef enum { FALSE = 0, TRUE } boolean;

typedef  union                      /* 1 ﾜｰﾄﾞの共用体定義 */
{
         struct  {                           /* 16 ﾋﾞｯﾄのﾌｨｰﾙﾄﾞ定義               */
                  unsigned char    b15 : 1 ;
                  unsigned char    b14 : 1 ;
                  unsigned char    b13 : 1 ;
                  unsigned char    b12 : 1 ;
                  unsigned char    b11 : 1 ;
                  unsigned char    b10 : 1 ;
                  unsigned char    b9 : 1 ;
                  unsigned char    b8 : 1 ;
                  unsigned char    b7 : 1 ;
                  unsigned char    b6 : 1 ;
                  unsigned char    b5 : 1 ;
                  unsigned char    b4 : 1 ;
                  unsigned char    b3 : 1 ;
```

```
                unsigned char      b2 : 1 ;
                unsigned char      b1 : 1 ;
                unsigned char      b0 : 1 ;
        } bi ;
        struct    {
            struct {                                  /* 上位ﾊﾞｲﾄのﾋﾞｯﾄﾌｨｰﾙﾄﾞ定義 */
                unsigned char      b7 : 1 ;
                unsigned char      b6 : 1 ;
                unsigned char      b5 : 1 ;
                unsigned char      b4 : 1 ;
                unsigned char      b3 : 1 ;
                unsigned char      b2 : 1 ;
                unsigned char      b1 : 1 ;
                unsigned char      b0 : 1 ;
            } h ;
            struct {                                  /* 下位ﾊﾞｲﾄのﾋﾞｯﾄﾌｨｰﾙﾄﾞ定義 */
                unsigned char      b7 : 1 ;
                unsigned char      b6 : 1 ;
                unsigned char      b5 : 1 ;
                unsigned char      b4 : 1 ;
                unsigned char      b3 : 1 ;
                unsigned char      b2 : 1 ;
                unsigned char      b1 : 1 ;
                unsigned char      b0 : 1 ;
            } l ;
        } bbi ;
        struct    {                                   /* ﾊﾞｲﾄ定義                 */
                unsigned char      h ;
                unsigned char      l ;
        } b ;
        unsigned short             h ;
} UShort ;

typedef   struct
{                                              /* ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀのﾋﾞｯﾄ構成    */
        unsigned char      FME       :1 ;      /* ﾌﾗｯｼｭﾒﾓﾘｲﾈｰﾌﾞﾙﾋﾞｯﾄ              */
        unsigned char      CE        :1 ;      /* ﾁｯﾌﾟｲﾈｰﾌﾞﾙﾋﾞｯﾄ                  */
        unsigned char      BSY       :1 ;      /* ﾋﾞｼﾞｰﾋﾞｯﾄ                       */
        unsigned char                :3 ;
        unsigned char      CLE       :1 ;      /* ｺﾏﾝﾄﾞﾗｯﾁｲﾈｰﾌﾞﾙﾋﾞｯﾄ              */
        unsigned char      ALE       :1 ;      /* ｱﾄﾞﾚｽﾗｯﾁｲﾈｰﾌﾞﾙﾋﾞｯﾄ              */
} TFmemCtl ;
```

```
typedef  union
{
    struct
    {                                   /* ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀｽﾃｰﾀｽのﾋﾞｯﾄ構成             */
        unsigned char      PRO      :1 ;    /* 0: Protect        1: Not protect    */
        unsigned char      BSY      :1 ;    /* 0: Busy           1: Ready          */
        unsigned char      SUS      :1 ;    /* 0: Not Suspended 1: Suspend         */
        unsigned char               :4 ;
        unsigned char      PAS      :1 ;    /* 0: Pass           1: Fail           */
    } bi ;
    unsigned char  b ;
} UFmemStatus ;

/*       関数プロトタイプ宣言        */
void main( void ) ;
void FmInit( void ) ;
void FmRead( unsigned short page, unsigned char *data ) ;
boolean FmWrite( unsigned short page, unsigned char *data ) ;
boolean FmErase( unsigned short block ) ;

/* 変数定義 */
#define   FmCntl    (*(TFmemCtl *)(0xFFF601))          /* ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀ        */
#define   FmCntl1   (*(TFmemCtl *)(0xFFF603))          /* ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀ 1      */
#define   FmData    (*(unsigned char *)(0xFFF604))     /* ﾌﾗｯｼｭﾒﾓﾘﾃﾞｰﾀﾊﾞｽ          */

                                    /* ﾌﾗｯｼｭﾒﾓﾘ動作ｺﾏﾝﾄﾞ                      */
#define   FM_AWRITE          0x80       /* ｼﾘｱﾙﾃﾞｰﾀ入力                       */
#define   FM_READ1           0x00       /* ﾘｰﾄﾞﾓｰﾄﾞ(1)                        */
#define   FM_READ2           0x50       /* ﾘｰﾄﾞﾓｰﾄﾞ(3)                        */
#define   FM_RESET           0xFF       /* ﾘｾｯﾄ                               */
#define   FM_PROG            0x10       /* ｵｰﾄﾌﾟﾛｸﾞﾗﾑ                         */
#define   FM_BERASE          0x60       /* ｵｰﾄﾌﾞﾛｯｸ消去 ｾｯﾄｱｯﾌﾟ               */
#define   FM_STATE           0x70       /* ｽﾃｰﾀｽﾘｰﾄﾞ                          */
#define   FM_RESUME          0xD0       /* ｵｰﾄﾌﾞﾛｯｸ消去 実行                  */

                        /* NAND ﾌﾗｯｼｭﾒﾓﾘ書込ﾃﾞｰﾀ             */
const unsigned char cstdat[3] = { 0x00, 0x55, 0xAA } ;

unsigned char wtbuff[528] ;         /* NAND ﾌﾗｯｼｭﾒﾓﾘ書込用ﾊﾞｯﾌｧ          */
unsigned char rdbuff[528] ;         /* NAND ﾌﾗｯｼｭﾒﾓﾘ読込用ﾊﾞｯﾌｧ          */
```

```
/*------------------------------------------------------------------
         メインルーチン
------------------------------------------------------------------*/
void main( void )
{
    FmInit() ;              /* NAND ﾌﾗｯｼｭﾒﾓﾘ初期設定            */

    while( 1 )
    {
        /*
          ８Mﾊﾞｲﾄの NAND ﾌﾗｯｼｭﾒﾓﾘ  8192ﾍﾟｰｼﾞの書込／読込
          1ﾍﾟｰｼﾞ=528ﾊﾞｲﾄ, 1ﾌﾞﾛｯｸ=16ﾍﾟｰｼﾞ,  全 512ﾌﾞﾛｯｸ
          全ﾒﾓﾘ書込／読み出し後、ﾌﾞﾛｯｸ単位で全ﾌﾞﾛｯｸ消去する。
        */
        for( i=0; i<8192; i++ )
        {
            dat = cstdat[i%3] ;             /* 書込ﾃﾞｰﾀ設定            */
            for( j=0; j<264; j++ )
            {
                wtbuff[j] = dat ;
            }
            FmWrite( i, wtbuff ) ;          /* ﾌﾗｯｼｭﾒﾓﾘ書込            */
            FmRead( i, rdbuff ) ;           /* ﾌﾗｯｼｭﾒﾓﾘ読み出し        */
            for( j=0; j<264; j++ )
            {
                if( wtbuff[j] != rdbuff[j] )
                    break ;
            }
        }
        /*
          ﾌﾞﾛｯｸ単位でﾌﾗｯｼｭﾒﾓﾘ消去 (全 512ﾌﾞﾛｯｸ)
        */
        for( j=0; j<512; j++ )
        {
            if( !FmErase( j ) )             /* ﾌﾗｯｼｭﾒﾓﾘ消去            */
                break ;
        }
        sleep() ;
    }
}

/*------------------------------------------------------------------
         NAND ﾌﾗｯｼｭﾒﾓﾘ初期設定
------------------------------------------------------------------*/
void FmInit( void )
{
    /*
      IOOSL ﾚｼﾞｽﾀの FME ﾋﾞｯﾄ ON にすることにより IOOD ﾚｼﾞｽﾀをﾌﾗｯｼｭﾒﾓﾘ制御用
      ﾚｼﾞｽﾀとして使用する。
    */
    FmCntl1.FME = 1 ;
}
```

```
/*----------------------------------------------------------------
         NAND ﾌﾗｯｼｭﾒﾓﾘ  1ﾍﾟｰｼﾞ  読み出し
         引数：page --- ﾍﾟｰｼﾞ番号(0～8192)
                    data --- 読込ﾃﾞｰﾀ格納領域先頭ﾎﾟｲﾝﾀ
----------------------------------------------------------------*/
void FmRead( unsigned short page, unsigned char *data )
{
UShort          *x ;
int             i ;
unsigned char   sts ;

    /*  ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄ OFF(ﾚﾃﾞｨ)ﾁｪｯｸ       */
    for( ;; ){
        if( FmCntl.BSY )
            break ;
    }
    /*  ﾍﾟｰｼﾞｱﾄﾞﾚｽ、ｺﾏﾝﾄﾞの設定      */
    x = (UShort *)&page ;
    FmCntl.ALE = 0 ;                /* ALE ﾋﾞｯﾄ OFF                     */
    FmCntl.CLE = 0 ;                /* CLE ﾋﾞｯﾄ OFF                     */
    FmCntl.CE  = 0 ;                /* CE  ﾋﾞｯﾄ OFF                     */

    FmCntl.CLE = 1 ;                /* CLE ﾋﾞｯﾄ ON                      */
    FmData = FM_READ1 ;             /* ﾘｰﾄﾞﾓｰﾄﾞ(1) ｺﾏﾝﾄﾞ ｾｯﾄ            */
    FmCntl.CLE = 0 ;                /* CLE ﾋﾞｯﾄ OFF                     */

    FmCntl.ALE = 1 ;                /* ALE ﾋﾞｯﾄ ON                      */
    FmData = 0 ;                    /* ｶﾗﾑｱﾄﾞﾚｽ A0 ～ A7 ｾｯﾄ            */
    FmData = x->b.l ;               /* ﾍﾟｰｼﾞｱﾄﾞﾚｽ A9 ～ A16 ｾｯﾄ         */
    FmData = x->b.h ;               /* ﾍﾟｰｼﾞｱﾄﾞﾚｽ A17 ～ A22 ｾｯﾄ        */
    FmCntl.ALE = 0 ;                /* ALE ﾋﾞｯﾄ OFF                     */
    sts = 0 ;

    /*  ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄ OFF(ﾚﾃﾞｨ)ﾁｪｯｸ       */
    for( ;; ){
        if( FmCntl.BSY )
            break ;
    }
    /*  1ﾍﾟｰｼﾞ分のﾃﾞｰﾀ読み出し       */
    for( i = 0; i < 264; i++ ){
        data[i] = FmData ;
    }
    /*  読み出し終了後、CE ﾋﾞｯﾄ ON             */
    FmCntl.CE = 1 ;
}
```

```
/*-----------------------------------------------------------------------
    NAND ﾌﾗｯｼｭﾒﾓﾘ  1ﾍﾟｰｼﾞ  書込
    引数：page ------ ﾍﾟｰｼﾞ番号(0～8192)
          data ------ 書込ﾃﾞｰﾀ領域先頭ﾎﾟｲﾝﾀ
          boolean --- 処理終了情報を返す。
                      正常終了：TRUE ， エラー終了：FALSE
    ノート：この関数はｵｰﾄﾍﾟｰｼﾞﾌﾟﾛｸﾞﾗﾑでﾌﾗｯｼｭﾒﾓﾘに書込を行っています。
            ﾃﾞｰﾀ入力ﾓｰﾄﾞにｾｯﾄし、ｱﾄﾞﾚｽ、ﾃﾞｰﾀを入力した後ﾌﾟﾛｸﾞﾗﾑｺﾏﾝﾄﾞ 10H
            を入力すると自動的にﾍﾟｰｼﾞﾌﾟﾛｸﾞﾗﾑが実行される。
-----------------------------------------------------------------------*/
boolean FmWrite( unsigned short page, unsigned char *data )
{
UShort          *x ;
int             i ;
UFmemStatus     sts ;
boolean         ret ;

    /*  ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄ OFF(ﾚﾃﾞｨ)ﾁｪｯｸ       */
    for( ;; ){
        if( FmCntl.BSY )
            break ;
    }
    /*  ﾍﾟｰｼﾞｱﾄﾞﾚｽ、ｺﾏﾝﾄﾞの設定      */
    x = (UShort *)&page ;
    FmCntl.ALE = 0 ;                /* ALE ﾋﾞｯﾄ OFF                     */
    FmCntl.CLE = 0 ;                /* CLE ﾋﾞｯﾄ OFF                     */
    FmCntl.CE  = 0 ;                /* CE  ﾋﾞｯﾄ OFF                     */

    FmCntl.CLE = 1 ;                /* CLE ﾋﾞｯﾄ ON                      */
    FmData = FM_AWRITE ;            /* ｼﾘｱﾙﾃﾞｰﾀ入力 ｺﾏﾝﾄﾞ ｾｯﾄ            */
    FmCntl.CLE = 0 ;                /* CLE ﾋﾞｯﾄ OFF                     */

    FmCntl.ALE = 1 ;                /* ALE ﾋﾞｯﾄ ON                      */
    FmData = 0 ;                    /* ｶﾗﾑｱﾄﾞﾚｽ A0 ～ A7 ｾｯﾄ            */
    FmData = x->b.l ;               /* ﾍﾟｰｼﾞｱﾄﾞﾚｽ A9 ～ A16 ｾｯﾄ         */
    FmData = x->b.h ;               /* ﾍﾟｰｼﾞｱﾄﾞﾚｽ A17 ～ A22 ｾｯﾄ        */
    FmCntl.ALE = 0 ;                /* ALE ﾋﾞｯﾄ OFF                     */

    /*  ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄ OFF(ﾚﾃﾞｨ)ﾁｪｯｸ       */
    for( ;; )
    {
        if( FmCntl.BSY )
            break ;

    }
```

```
    /*  1ﾍﾟｰｼﾞ分のﾃﾞｰﾀ書込 */
    for( i = 0; i < 264; i++ )
    {
        FmData = data[i] ;
    }
    FmCntl.CLE = 1 ;                  /* CLE ﾋﾞｯﾄ ON                        */
    FmData = FM_PROG ;                /* ｵｰﾄﾌﾟﾛｸﾞﾗﾑ ｺﾏﾝﾄﾞ ｾｯﾄ                */
    FmCntl.CLE = 0 ;                  /* CLE ﾋﾞｯﾄ OFF                       */

    /*  ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄ OFF(ﾚﾃﾞｨ)ﾁｪｯｸ        */
    for( ;; )
    {
        if( FmCntl.BSY )
            break ;
    }
    /* ﾃﾞｰﾀ書込終了確認ﾁｪｯｸ */
    for( ;; )
    {
        FmCntl.CLE = 1 ;              /* CLE ﾋﾞｯﾄ ON                        */
        FmData = FM_STATE ;           /* ｽﾃｰﾀｽﾘｰﾄﾞ ｺﾏﾝﾄﾞ ｾｯﾄ                 */
        FmCntl.CLE = 0 ;              /* CLE ﾋﾞｯﾄ OFF                       */
        sts.b = FmData ;              /* ｽﾃｰﾀｽ ﾃﾞｰﾀ 入力                    */

        if( !sts.bi.BSY )             /* BSY ﾋﾞｯﾄ ON(ﾚﾃﾞｨ) ?                */
            continue ;
        /* BSY ﾋﾞｯﾄ Ready のとき Pass/Fail ﾁｪｯｸ         */
        if( !sts.bi.PAS )
            ret = TRUE ;              /* PAS ﾋﾞｯﾄ ON(Fail)で正常終了        */
        else
            ret = FALSE ;             /* PAS ﾋﾞｯﾄ OFF(Pass)で異常終了       */
        break ;
    }
    /* 書込終了後、CE ﾋﾞｯﾄ ON                  */
    FmCntl.CE  = 1 ;
    return ret ;
}
```

```
/*-------------------------------------------------------------------
    NAND ﾌﾗｯｼｭﾒﾓﾘ  1ﾌﾞﾛｯｸ  消去
    引数：block ----- ﾌﾞﾛｯｸ番号(0～512)
          boolean --- 処理終了情報を返す。
                      正常終了：TRUE , エラー終了：FALSE
    ノート：この関数はｵｰﾄﾌﾞﾛｯｸ消去でﾌﾗｯｼｭﾒﾓﾘをﾌﾞﾛｯｸ単位で消去します。
            ﾌﾞﾛｯｸ消去ｾｯﾄｱｯﾌﾟｺﾏﾝﾄﾞ 60H 入力後、ﾌﾞﾛｯｸｱﾄﾞﾚｽを 2 ｻｲｸﾙでｾｯﾄし
            消去実行ｺﾏﾝﾄﾞ D0H を入力するとﾌﾞﾛｯｸ消去動作が開始されます。
-------------------------------------------------------------------*/
boolean FmErase( unsigned short block )
{
UShort          xx ;
UShort          *x ;
UFmemStatus     sts ;
boolean         ret ;

    /* ﾌﾞﾛｯｸｱﾄﾞﾚｽは A13～A22 なので 4 ﾋﾞｯﾄ左にｼﾌﾄする      */
    x = (UShort *)&xx ;
    xx = block << 4 ;

    FmCntl.ALE = 0 ;             /* ALE ﾋﾞｯﾄ OFF            */
    FmCntl.CLE = 0 ;             /* CLE ﾋﾞｯﾄ OFF            */
    FmCntl.CE  = 0 ;             /* CE  ﾋﾞｯﾄ OFF            */

    /*  ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄ OFF(ﾚﾃﾞｨ)ﾁｪｯｸ        */
    for( ;; ){
        if( FmCntl.BSY )
            break ;
    }

    /*  ﾍﾟｰｼﾞｱﾄﾞﾚｽ、ｺﾏﾝﾄﾞの設定     */
    FmCntl.CLE = 1 ;             /* CLE ﾋﾞｯﾄ ON                       */
    FmData = FM_BERASE ;         /* ｵｰﾄﾌﾞﾛｯｸ消去ｾｯﾄｱｯﾌﾟ ｺﾏﾝﾄﾞ ｾｯﾄ     */
    FmCntl.CLE = 0 ;             /* CLE ﾋﾞｯﾄ OFF                      */

    FmCntl.ALE = 1 ;             /* ALE ﾋﾞｯﾄ ON                       */
    FmData = x->b.l ;            /* ﾌﾞﾛｯｸｱﾄﾞﾚｽ A13 ～ A16 ｾｯﾄ         */
    FmData = x->b.h ;            /* ﾌﾞﾛｯｸｱﾄﾞﾚｽ A17 ～ A22 ｾｯﾄ         */
    FmCntl.ALE = 0 ;             /* ALE ﾋﾞｯﾄ OFF                      */

    FmCntl.CLE = 1 ;             /* CLE ﾋﾞｯﾄ ON                       */
    FmData = FM_RESUME ;         /* ｵｰﾄﾌﾞﾛｯｸ消去実行 ｺﾏﾝﾄﾞ ｾｯﾄ         */
    FmCntl.CLE = 0 ;             /* CLE ﾋﾞｯﾄ OFF                      */

    /*  ﾌﾗｯｼｭﾒﾓﾘ制御ﾚｼﾞｽﾀの BSY ﾋﾞｯﾄ OFF(ﾚﾃﾞｨ)ﾁｪｯｸ        */
    for( ;; ){
        if( FmCntl.BSY )
            break ;
    }
```

```
    /*  ﾌﾞﾛｯｸ消去終了確認ﾁｪｯｸ          */
    for( ;; )
    {
        FmCntl.CLE = 1 ;             /* CLE ﾋﾞｯﾄ ON                    */
        FmData = FM_STATE ;          /* ｽﾃｰﾀｽﾘｰﾄﾞ ｺﾏﾝﾄﾞ ｾｯﾄ             */
        FmCntl.CLE = 0 ;             /* CLE ﾋﾞｯﾄ OFF                   */
        sts.b = FmData ;             /* ｽﾃｰﾀｽ ﾃﾞｰﾀ 入力                */

        if( !sts.bi.BSY )            /* BSY ﾋﾞｯﾄ ON(ﾚﾃﾞｨ) ?            */
            continue ;
        /* BSY ﾋﾞｯﾄ Ready のとき Pass/Fail ﾁｪｯｸ          */
        if( !sts.bi.PAS )
            ret = TRUE ;             /* PAS ﾋﾞｯﾄ ON(Fail)で正常終了    */
        else
            ret = FALSE ;            /* PAS ﾋﾞｯﾄ OFF(Pass)で異常終了   */
        break ;
    }
    /*  ﾌﾞﾛｯｸ消去終了後、CE ﾋﾞｯﾄ ON     */
    FmCntl.CE = 1 ;
    return ret ;
}
```

第８章　プログラミングの方法

　通常のＲＯM化する方法を用いコンパイル等を実行してください。プログラムの転送はモトローラＳフォーマットしか受け付けません。モトローラＳフォーマットのレコードタイプＳ１、Ｓ２、Ｓ３のアドレスモードに対応しています。

ＡＴ８０１０のユーザーモード２～5を使用する場合、Ｈ８の１６Mアドレスモードになります。このモードの場合、初期設定でＡ20～Ａ23はアドレス出力にはなっていませんのでＨ８のポート設定レジスタでアドレス出力に指定してください。

第９章　電気的特性

Ｈ８マイコンから直接出力される端子はＨ８マイコンの特性を参照してください。

■ 絶対最大定格

項　目	記 号	定　格　値	単 位
電源電圧	Vcc	-0.3 ～ +7.0	V
入力電圧	Vin	-0.3 ～ Ｖｃｃ+0.3	V
動作温度		-20 ～ +70	℃
保存温度		-30 ～ +90	℃

■ ＤＣ特性

H8 マイコン直結端子及び、32KOUT、CLKC 端子を除く

項　目	記 号	Min	Typ	Max	単 位
入力 High レベル電圧	VIH	2.0		Vcc+0.3	V
入力 Low レベル電圧	VIL	-0.3		0.8	V
出力 High レベル電圧（IOH＝-6mA）	VOH	3.84			V
出力 Low レベル電圧（IOL＝6mA）	VOL			0.33	V

■ ＡＣ特性

Ｈ８マイコンの仕様書を参照してください。また、クロックは９．８３０４ＭＨｚを使用しております。

第１０章　評価ボード

　ＡＴ８０１０の評価、実験を行う為の評価ボードを用意しております。
ボードサイズは１１４×２５０ミリで、**ＡＴ８０１０**の全端子を２．５４ピッチに変換したユニバーサル基板となっています。